

02/6/06

22W  
AFF

PTO/SB/21 (09-04)

Approved for use through 07/31/2006. OMB 0651-0031

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

## TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission

Application Number	10/021,521
Filing Date	10/29/2001
First Named Inventor	Gary R. Maze
Art Unit	2161
Examiner Name	Etienne Pierre LeRoux
Attorney Docket Number	E4507-2

### ENCLOSURES (Check all that apply)

<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment/Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement  <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Reply to Missing Parts/ Incomplete Application <input type="checkbox"/> Reply to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Drawing(s) <input type="checkbox"/> Licensing-related Papers  <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____ <input type="checkbox"/> Landscape Table on CD	<input type="checkbox"/> After Allowance Communication to TC  <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input checked="" type="checkbox"/> Appeal Communication to TC (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Other Enclosure(s) (please identify below): Acknowledgment Postcard
<div>Remarks</div> <div>BEST AVAILABLE COPY</div>		

### SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Duane Morris LLP		
Signature			
Printed name	Gary R. Maze		
Date	02/03/2006	Reg. No.	42,851

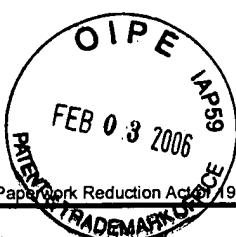
### CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature			
Typed or printed name	Tracie Thigpen	Date	02/03/2006

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



PTO/SB/17 (01-06)

Approved for use through 07/31/2006. OMB 0651-0032

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

**FEE TRANSMITTAL**  
**For FY 2006**☒ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT (\$) 250

**Complete if Known**

Application Number	10/021,521
Filing Date	10/29/2001
First Named Inventor	Gary R. Maze
Examiner Name	Etienne Pierre LeRoux
Art Unit	2161
Attorney Docket No.	E4507-2

**METHOD OF PAYMENT (check all that apply)**☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): \_\_\_\_\_☒ Deposit Account Deposit Account Number: 02-0429 Deposit Account Name: Duane Morris LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, except for the filing fee☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments**WARNING:** Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**FEE CALCULATION (All the fees below are due upon filing or may be subject to a surcharge.)****1. BASIC FILING, SEARCH, AND EXAMINATION FEES**

Application Type	FILING FEES		SEARCH FEES		EXAMINATION FEES		Fees Paid (\$)
	Fee (\$)	Small Entity Fee (\$)	Fee (\$)	Small Entity Fee (\$)	Fee (\$)	Small Entity Fee (\$)	
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

**2. EXCESS CLAIM FEES****Fee Description**

Each claim over 20 (including Reissues)

Each independent claim over 3 (including Reissues)

Multiple dependent claims

Fee (\$)	Small Entity Fee (\$)
50	25
200	100
360	180

Total Claims	Extra Claims	Fee (\$)	Fee Paid (\$)
--------------	--------------	----------	---------------

- 20 or HP = \_\_\_\_\_ x \_\_\_\_\_ = \_\_\_\_\_

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims	Extra Claims	Fee (\$)	Fee Paid (\$)
---------------	--------------	----------	---------------

- 3 or HP = \_\_\_\_\_ x \_\_\_\_\_ = \_\_\_\_\_

HP = highest number of independent claims paid for, if greater than 3.

**Multiple Dependent Claims**

Fee (\$)	Fee Paid (\$)
----------	---------------

**3. APPLICATION SIZE FEE**

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets	Extra Sheets	Number of each additional 50 or fraction thereof	Fee (\$)	Fee Paid (\$)
--------------	--------------	--	----------	---------------

- 100 = \_\_\_\_\_ / 50 = \_\_\_\_\_ (round up to a whole number) x \_\_\_\_\_ = \_\_\_\_\_

**4. OTHER FEE(S)**

Non-English Specification, \$130 fee (no small entity discount)

Other (e.g., late filing surcharge): Filing fee for Brief in Support of Appeal [41.20(b)(2)]**Fees Paid (\$)**

250

**SUBMITTED BY**

Signature

Registration No.  
(Attorney/Agent) 42,851

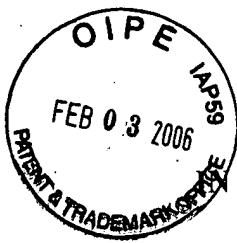
Telephone 713.402.3917

Name (Print/Type) Gary R. Maze

Date 02/03/2006

This collection of information is required by 37 CFR 1.136. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 30 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Maze, <i>et al.</i>	§	Group Art Unit:	2171
Serial No:	10/021,521	§		
		§	Examiner:	LeRoux, E.
Filed:	October 29, 2001	§		
		§	Attorney Docket:	E4507-002
For:	System And Method For The	§		
	Management Of Distributed	§		
	Personalized Information	§		

Mail Stop Appeal Brief Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**APPELLANT'S BRIEF**

**I. Real Party in Interest**

The real parties in interest are Gary R Maze and Dana L Schnitzer.

**II. Related Appeals and Interferences**

There are no related appeals or interferences.

**III. Status of Claims**

Claims 1-20 of the application under appeal (Serial No. 10/021,521 (the "521 Application")) are pending and stand finally rejected.

Rejection of all claims is being appealed.

**IV. Status of Amendments**

In response to the Final Office Action, on November 10, 2005 Applicants supplied proposed amendments to the specification to include a summary and correct typographical errors. Applicants further supplied proposed amendments to the claims.

All Amendments filed on November 10, 2005 after the Final Office Action have not been entered as indicated in the Advisory Action dated December 12, 2005.

## V. Summary Of Claimed Subject Matter

The inventions claimed comprise a system useful for organization of information. In the claimed inventions, the system comprises a database comprising (a) an authority table which contains "authorities," defined to mean raw data or links to the raw data; (b) at least one taxonomy table (containing categories) that is manipulatable by a user with adequate access permission to manipulate the user taxonomy table; and (c) at least one user manipulatable summary table that is related to the authority table and user taxonomy table. In other claimed inventions, methods of creating summarized information for later access for a system claimed in the claims of the '521 Application comprise capturing a description of raw data into the authority table; retrieving raw data from the authority table by a user; summarizing at least a portion of the raw data; examining one or more elements in a user taxonomy table for an appropriate taxonomy element to associate with the summarization and creating a new element in the user taxonomy table for an appropriate taxonomy element to associate with the summarization if an appropriate element is not already present in the taxonomy table; associating the summarization with the appropriate element of the taxonomy table; and storing the summarization with its associated element of the taxonomy table in the summary table. (*See, generally, '521 Application, ¶¶ [0023-0034]*)

More specifically with regard to the rejected independent claims, claim 1 claims a system for organization of information, comprising a server comprising a searchable authority table; a data communications device operatively in communication with the server; a user computer operatively in communication with the data communications device, the user computer having access to the searchable authority table, the user

computer further comprising at least one user definable taxonomy table, wherein the taxonomy table is accessible to the user computer and is manipulatable by a user with adequate access permission to manipulate the user taxonomy table; and at least one searchable summary table, wherein the summary table is accessible to the user computer, is related to the authority table and the user taxonomy table and is manageable by a user with adequate access permission to manage the summary table. (*See, generally*, '521 Application, ¶¶ [0023-0034]) The system additionally comprises software executable in the server to provide access to and management of the authority table and software executable in the user computer to provide access to and manipulation of the taxonomy table and the summary table. (*See, generally*, '521 Application, ¶¶ [0023-0034])

In the two independent method claims (claims 12 and 17), summarized information is created for later access for a system of claim 1 by capturing raw data into a programmatically manipulatable format; storing the programmatically manipulatable format of the raw data into the authority table; capturing a description of a source of the raw data; storing the description into the authority table while associating the description with the data in the authority table; programmatically retrieving the raw data from the authority table by a user; allowing the user to summarize at least a portion of the retrieved raw data, the summarization to be stored in the summary table; examining one or more elements in the user taxonomy table for an appropriate taxonomy table element to associate with the summarization; creating a new element in the user taxonomy table describing an appropriate taxonomy element to associate with the summarization if an appropriate element is not already present in the taxonomy table; associating the summarization with the appropriate element of the taxonomy table; associating the

summarization with the authority table; and storing the summarization and associations in the summary table. (See, generally, '521 Application, ¶¶ [0067] and Figure 8) In claim 17, a system of claim 1 may be searched for relevant information by formulating a query at a user workstation; analyzing the query for keywords; obtaining user filtering input for tables to be searched; searching for keywords against the tables using the user filtering input; and returning search results to the user. (See, generally, '521 Application, ¶¶ [0067] and Figure 8)

Figures 1-7 are illustrative schematic views of exemplary embodiments of the system of the '521 Application, including a schematic object-like view of organization of table elements of an exemplary embodiment of the system of the '521 Application as well as exemplary taxonomy screen forms such as might be presented to an end user of the system. Figures 8a and 8b are flowcharts of an exemplary embodiment of the system of the inventions claimed in the '512 Application.

There are no means-plus-function limitations.

## **VI. Grounds Of Rejection To Be Reviewed On Appeal**

The following grounds of rejection are presented for review:

1. Claims 1-20 were rejected as being invalid under 35 U.S.C. §112, ¶ 1 for an alleged failure to contain a full, clear and concise description of the limitations in those claims. (First Office Action, pages 4-8; Final Office Action, pages 2-3)
2. Claims 12-14 and 16-20 were rejected under 35 U.S.C. §112 ¶ 1 as not being enabled. (First Office Action, page 9; Final Office Action, page 3)
3. Claims 1 and 4-11 were rejected as being anticipated under 35 U.S.C. §102(e) by Chakrabarti. (First Office Action, pages 9-12; Final Office Action, pages 3-6)

4. Claim 2 was rejected as being rendered obvious under 35 U.S.C. §103(a) by Chakrabarti in view of Price. (First Office Action, pages 12-13; Final Office Action, page 7)

5. Claim 3 was rejected as being rendered obvious under 35 U.S.C. §103(a) by Chakrabarti alone. (First Office Action, pages 13-14; Final Office Action, pages 7-8)

## **VII. Argument**

### **A. Claims 1-20 Are Enabled Under 35 U.S.C. §112, ¶ 1**

#### **1. The §112 Rejections**

In the First Office Action dated October 24, 2004 ("First Office Action"), the Patent Examiner asserted that certain claims were invalid under 35 U.S.C. §112 ¶ 1, asserting that the "specification does not contain a full, clear and concise description [of these claimed inventions] such that a skilled artisan can make or use the present invention," specifically asserting that the following were "unclear:"

- \* in claim 1, (a) generation of the summary table which the Patent Examiner stated is solely based on user preference and is not disclosed in the specification and (b) the relationship between the summary table, the authority table, and the taxonomy table;
- \* in claim 12, (a) the method generating a summary table; (b) the meaning of "appropriate taxonomy table element;" and (c) how to accomplish the "above [sic] (1) associating, (2) summarization, and (3) appropriate element"
- \* in claims 13 and 14, the method of parsing the raw data to produce keywords

- \* in claim 16, the meaning or method of (a) receiving notice, (a) receiving keywords, (c) obtaining keywords, (d) examining keywords, (e) generating a relevance factor, and (f) “suggesting”
- \* in claim 17, the meaning or method of (a) formulating a query, (b) obtaining user filtering, and (c) user filtering input
- \* in claim 18, the methods of (a) searching outside the tables and (b) searching below a predetermined threshold
- \* in claim 19, the meaning or method of limiting searches
- \* in claim 20, the method of viewing

The Patent Examiner further rejected Claim 15 under 35 U.S.C. §112 ¶ 1, asserting that it was not possible to determine the scope of the claim because it was allegedly unclear “how a description of the raw data can comprise the raw data [itself].”

The Patent Examiner then refused to provide an art rejection of claims 12-14 and 16-20 because the Patent Examiner asserted that these claims failed to comply with 35 U.S.C. §112 ¶ 1 but did not specify any further reason(s) for such a statement. (First Office Action, page 9)

After timely response by Applicants, in the Final Office Action dated October 3, 2005 (“Final Office Action”), the Patent Examiner again rejected claims 1 and 12 under 35 U.S.C. §112 ¶ 1, this time on the apparent sole basis of an asserted lack of full, clear and concise description of the process of relating the authority table to the taxonomy table. (Final Office Action, page 3) Claims 2-11 and 13-20 stood rejected as allegedly depending from a rejected base claim. (Final Office Action, page 3) Importantly, this is *prima facie* error because claim 17 is an independent claim and claims 18-20 depend



from claim 17.

Once again, the Patent Examiner refused to provide an art rejection of claims 12-14 and 16-20 because the Patent Examiner asserted that these claims failed to comply with 35 U.S.C. §112 ¶ 1 but again did not specify any further reason(s) for such a statement. (Final Office Action, page 3)

It is thus unclear but arguable that the Patent Examiner did not maintain all of his prior rejections under 35 U.S.C. §112 ¶1 from the First Office Action. Applicants nonetheless traverse all of the Patent Examiner's rejections and objections.

## **2. The Claims Are Enabled As A Matter Of Law**

The Federal Circuit has admonished against including in the specification material that is known in the art. *Spectra-Physics, Inc. v. Coherent, Inc.*, 827 F.2d 1524, 1534 (Fed. Cir. 1987) ("A patent need not teach, and preferably omits, what is well known in the art."). Additionally, an applicant is not required to describe in the specification every conceivable and possible future embodiment of his invention. *SRI Int'l v. Matsushita Elec. Corp. of America*, 775 F.2d 1107, 1121 (Fed. Cir. 1985) (en banc). In fact, requiring inventors to include every imaginable detail, including those widely understood by persons of ordinary skill in the art, would be the antithesis of conciseness and would result in exceedingly lengthy patents. *Atmel Corporation v. Information Storage Devices, Inc.*, 198 F.3d 1374 (Fed. Cir. 2001). Accordingly, a patent applicant should not include in the specification that which is already known to and available to the public. *Paperless Accounting, Inc. v. Bay Area Rapid Transit System*, 804 F.2d 659 (Fed. Cir. 1986).

As opposed to the Patent Examiner's proffered subjective opinions (*see, e.g.*, Final Office Action, page 3), the actual test for sufficiency of support in an application is

whether the disclosure of the application relied upon “reasonably conveys to the artisan that the inventor had possession at that time of the later claimed subject matter.” *Ralston Purina Company v. Far-Mar-Co, Inc.*, 772 F.2d 1570 (Fed. Cir. 1985). The MPEP states:

Definiteness of claim language must be analyzed, not in a vacuum, but in light of:

- (A) The content of the particular application disclosure;
- (B) The teachings of the prior art; and
- (C) The claim interpretation that would be given by one possessing the ordinary level of skill in the pertinent art at the time the invention was made.

(2173.02 Clarity and Precision [R-3]) In reviewing a claim for compliance with 35 U.S.C. § 112, second paragraph, “the examiner must consider the claim as a whole to determine whether the claim apprises one of ordinary skill in the art of its scope and, therefore, serves the notice function required by 35 U.S.C. ¶ 112, second paragraph, by providing clear warning to others as to what constitutes infringement of the patent.” (*Id.*)

As proven below, each of the phrases and limitations to which the Patent Examiner objected has enabling support in the specification’s text and figures when coupled with knowledge of the meaning of terms in 2001 to one of ordinary skill in these arts. Each of these rejections is fully traversed below.

**a) Relational Databases And Their Structures Are Not New**

Discussing the then-state-of-the-art with respect to U.S. Patent No. 5,555,403, which issued September 10, 1996, the Federal Circuit explained that

A relational database is a computerized compilation of data organized into tables, each table having columns (attributes), with column headings, and rows of information. **Tables that share at least one attribute in common are “related.” Tables without a common attribute may still be related via other tables with which they do share a common attribute.** The pathways relating those separate tables to each other are called “joins.” **Once tables have been related by a join, a user may combine or**

**correlate the information in the joined tables to derive new useful information.**

Users access and correlate information in relational databases only by use of a relational database management system (RDBMS), which consists of hardware and software. **To access such information, a user sends queries to the RDBMS, which executes the queries and retrieves the requested information from the tables in the relational database.** A RDBMS, however, only recognizes queries written in complex “query languages.” The most common query language is Structured Query Language (SQL).

**A proper query in these languages consists of one or more “clauses.”** Common types of clauses are SELECT, WHERE, FROM, HAVING, ORDER BY, and GROUP BY clauses. Thus, to compose a proper inquiry, a user must understand the structure and content of the relational database as well as the complex syntax of the specific query language. These complexities generally prevent laypersons from drafting queries in query languages.

*Business Objects, S.A. v. Microstrategy, Inc.*, 393 F.3d 1366, 1368 (Fed. Cir 2005).<sup>1</sup>

Thus, relational databases, their structures, and their methods of relating tables and of use, e.g. queries, were well known in the art by October 2001, the date of filing of the ‘521 Application on appeal.

**b) Claim 1**

Claim 1 requires that “the user computer further compris[es] ... at least one searchable summary table.”<sup>2</sup> (‘521 Application, claim 1) In 2001, a person of ordinary skill in the computer arts, especially in the database arts, would have readily understand that a “searchable summary table” in Claim 1 is a table –a widely, commonly understood database term in October 2001 – which can be searched, i.e. using software. *Business Objects, S.A.*, 393 F.3d at 1368.<sup>3</sup>

---

<sup>1</sup> (See also Exhibit A, Ullman, PRINCIPLES OF DATABASE SYSTEMS (1980))

<sup>2</sup> As was well known in 2001, tables may exist as part of databases or as standalone tables.

<sup>3</sup> (Exhibit A, Ullman, PRINCIPLES OF DATABASE SYSTEMS, page 81 (“Interrogation of a Relational Database”))

The Patent Examiner evidently confused generation of data with generation of a table. Claim 1 clearly does not require generation of the summary table (i.e., its structure) based on user preference as the Patent Examiner asserted. (First Office Action, page 4) Indeed, the specification notes that “database 22 may comprise user manipulatable data, referred to herein as ‘summary’ data, typically in summary table 12a associated with computer 10.” (‘521 Application, ¶ [0023] (emphasis added)). Clearly, the summary table contains data that a user may manipulate but requires nothing with respect to the user creating the table itself.<sup>4</sup>

Claim 1 further requires that the summary table “is related to the authority table and the user taxonomy table.” The Patent Examiner asserts that one of ordinary skill would not know how to relate the summary table to the authority and taxonomy tables. (Final Office Action, pages 10-11) By 2001, persons of ordinary skill in database arts recognized “related” as a term of art – in fact, relational databases were old art by 2001. *Business Objects, S.A.*, 393 F.3d at 1368<sup>5</sup>; Exhibit A, Ullman, PRINCIPLES OF DATABASE SYSTEMS, pages 83-86. Moreover, methods of, and structures for, relating tables, including the term “one-to-many,” were well known in the art at the time of filing of the current application. See, e.g., *Id.*; Exhibit A, Ullman, page 77; Exhibit B, <http://www.campus.ncl.ac.uk/databases/design/design.html> (original copyright 1997)) It was also well known to associate tables to each other in a database using keys, i.e. fields or combinations of fields. (See, e.g., *Id.*; Exhibit A, Ullman, pages 76, 86) The ‘521

---

<sup>4</sup> A person of ordinary skill in the computer arts in October 2001 would recognize that a database software and/or a database administrator may give a user only read access. The user can still manipulate the table and its data, e.g. search for and read a record’s data, but may not change the data. (Exhibit A, Ullman, PRINCIPLES OF DATABASE SYSTEMS, page 80 (“Operations on Relational Databases”))

<sup>5</sup> All references herein to *Business Objects, S.A.* are merely to indicate that the Federal Circuit recognized that what the Patent Examiner states without any evidence is unsupportable and that the Patent Examiner’s assertion of lack of clarity is baseless.

Specification teaches use of key fields in various embodiments. (See, e.g., ‘521 Application, ¶ [0023] (“In the preferred embodiment, each entry in each authority table 22a, described herein as an “authority, will have a unique identifier such as a primary key value.”); ¶ [0047] (“Linkage between backend portion 200 and user portion 300 may be by a unique key identifier such as to a unique authority table key value.”))

Therefore, as illustrated in the specification, one of ordinary skill in the art in October 2001 would thus certainly know that all tables can be related to one or more other tables, e.g. summary tables to authority and taxonomy tables, such as by using key fields. (See, e.g., ‘521 Application, Fig. 2 (showing exemplary relations between exemplary tables); Exhibit A, Ullman, page 85 (showing an entity-relationship diagram similar to Fig. 2))

Importantly, as noted above, the ‘521 Application, including its figures, refers to and describes such processes and structures, including graphically. For example:

Paragraph [0047] teaches “**Linkage** between backend portion 200 [which comprises the authority table] and user portion 300 [which comprises the summary table] may be by a **unique key identifier** such as to a **unique authority table key value**.” See, e.g., ‘521 Application, ‘Fig. 2

Paragraph [0044] teaches that “User summary table 12b may be part of user portion 300 and contains **an element for each item of information the user desires to link to at least one element in user taxonomy table 12a**.” Thus, summary and taxonomy tables are taught in an embodiment as linked, i.e. related, by key fields.

Paragraph [0028] teaches that “In the preferred embodiment, each entry in each authority table 22a, described herein as an ‘authority,’ will have **a unique identifier such as a primary key value**.”

Describing an embodiment, paragraph [0044] teaches “**One currently envisioned method of relating user summary table 12b to authority table 22a is to use an intermediate table such as user knowledge table 12e which relates one element in authority table 22a to one element in user summary table 12b, although one-to-many relationships may be**

**defined in this or other manners as well.**<sup>6</sup>

Accordingly, relationships between these tables is fully, clearly, and concisely described such that one of ordinary skill in the art could practice the claimed inventions as of the filing date of the '521 Application and provide "clear warning to others as to what constitutes infringement of the patent." MPEP 2173.02. Claim 1 is thus fully, clearly, and concisely described and satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner should be reversed.

**c) Claim 12**

Claim 12 claims a method for creating summarized information for later access for a system of claim 1. Limitation 12(f) makes clear that the user is allowed to summarize the authority's raw data, i.e. put someone else's language existent in a "raw data" document such as a case or statute or picture into words meaningful to the user, and that this summarization created by the user is that which is stored in the summary table. (See, e.g., '521 Application, Fig. 4 (showing an exemplary form for entry of a summary)) The Patent Examiner evidently confuses generation of the summary table itself with generation of the data populating that table. There is nothing in claim 12 which requires users to create any table, i.e. to create the table structure which will be used to hold data. (See, e.g., '521 Application Figs. 3 and 8 (where creating taxonomy data is shown via use of a form to enter the taxonomy data))

A "taxonomy element," i.e. its data, is also referred to in the '521 Application as a "category." ('521 Application ¶ [0043] and Fig. 3) "User taxonomy table 12a may be part of user portion 300 and contains user defined categorizations as well as

---

<sup>6</sup> On page 10 of the Final Office Action, the Patent Examiner confuses the summary table (which contains user data that the user created to summarize an authority in the user's own words) with relations between tables. This is addressed later in this brief.

relationships between each member of taxonomy table 12a to others in taxonomy table 12a.” (*Id.*; *see, e.g.*, ‘521 Application Fig. 2) The user may additionally associate the entered and captured summary with one or more elements in taxonomy table 12a. (*Id.*, at ¶ [0053] “Optionally, **the user may create new elements in user taxonomy table 12a for use with the summary;**” *see also* ¶ [0054] and Figure 4 (“Summary information may be shown such as at 420 and taxonomy elements associated with that summary for the authority may additionally be shown such as at 430.”)) With respect to limitation 12(g), the ‘521 Application teaches that in various embodiments an “appropriate taxonomy element” in context means something the user thinks appropriate to associate with that summary. (*See, e.g., Id.*, at ¶ [0053] (“The user may additionally associate the captured summary with one or more elements in taxonomy table 12a.”)) Thus, contrary to the Patent Examiner’s arguments on page 11 of the Final Office Action, one of ordinary skill in these arts does not need to know what the user thinks is appropriate. The claimed method requires only that a user is to be allowed to associate a taxonomy table element with the summary. The ‘521 Application teaches that these associations may be accomplished from the user’s perspective by (1) a user via screen forms, (2) database code, (3) other code, or (4) a combination thereof, as was the norm in 2001 in these arts. (*See, e.g.*, ‘521 Application, ¶[0053] and ¶[0061] and Figs. 3-7) For example, in embodiments one or more forms may be used to initiate the required associations. (*See, e.g.*, ‘521 Application, Figs. 3, 4, and 8) Once that association is initiated, the ‘521 Application teaches that the taxonomy table (hence, its elements which are categories) may be related to the authority and/or summary tables, e.g. via key fields in each table. (*See, e.g.*, ‘521 Application Fig. 2) Such processes were well known in the art in 2001,

as discussed herein above.

Thus, claim 2 is fully, clearly, and concisely described and enabled by the specification. Accordingly, claim 2 satisfies 35 U.S.C. §112 ¶ 1 and the and the Patent Examiner should be reversed.

**d) Claims 13 and 14**

Claims 13 and 14 recite “parsing the raw data prior to storing the captured raw data in the authority table.” The ‘521 Application teaches that data may be “parsed according to any of a number of equivalent methods to **tokenize or otherwise summarize the text file.**” (‘512 Application, ¶ [0029]) “Parse” was a well-known term of art in software in October 2001. (Exhibit C, [www.dictionary.com](http://www.dictionary.com), citing *The American Heritage® Dictionary of the English Language, Fourth Edition Copyright © 2000 by Houghton Mifflin Company* (“**Computer Science.** To analyze or separate (input, for example) into more easily processed components.”))

As was also well known in these arts in 2001, a keyword is “a word used as a reference point for finding other words or information.” (*Id.*) Therefore, contrary to the Patent Examiner’s arguments on page 11 of the Final Office Action, parsing raw data to create keywords was fully, clearly, and concisely stated and understandable to one of ordinary skill in the database arts to mean tokenize or otherwise summarize the text portion of the raw data to create words used as reference points for finding other words or information. This is consistent with the teachings of the specification, e.g. the ‘521 Application at ¶ [0029] teaches that for an embodiment:

By way of example and not limitation, a text file is captured into authority table 22a and **may be parsed according to any of a number of equivalent methods to tokenize or otherwise summarize the text file. Each token or summary identifier identified may be captured into an**



**element in an associated authority keyword table 22b where the element in that table 22b for that token comprises a word offset from a given, known starting point, e.g. the beginning of the text document, where that token exists.** By way of further example and not limitation, a user may arbitrarily associate one or more authority keywords with a file captured into or identified in an authority table 22a, such as a geographic location in a graphic file or a time sequence identifier for audio or audiovisual data. Additionally, in some currently envisioned alternative embodiments, authorities may be translated into a different language upon or after capture. Additionally still, the translation may occur on the fly using any one of numerous equivalent methods or software as will be familiar to those of ordinary skill in the software arts and translations may be kept in authority table 22a and linked to each other to facilitate use.

Thus, claims 13 and 14 are fully, clearly, and concisely described. Accordingly, claims 13 and 14 satisfy 35 U.S.C. §112 ¶ 1 and the Patent Examiner should be reversed.

**e) Claim 16**

Data communication between computers in a network was well known in these arts in 2001 and accomplishable in numerous, equivalent ways. The '521 Application illustrates a computer network ('521 Application Figs. 1 and 2) and states that "By way of example and not limitation, software executing at user computer 10,30,50 (FIG. 1) may receive notice that a new entry has been added to authority table 22a (FIG. 2)." ('521 Application ¶ [0063]) As would have been familiar to those of ordinary skill in these arts in 2001, "notification" using software may have been accomplished in numerous, equivalent ways, e.g. semaphores or TCP/IP packet events or e-mail protocol packets. Additionally, the '521 Application notes:

That software may gather the keywords associated with the new element of authority table 22a (FIG. 2) and then, using the user's taxonomy table 12a (FIG. 2), gather keywords associated by the user with that user taxonomy table 12a (FIG. 2) such as on an element by element basis. For each element in the taxonomy, the software can determine if those keywords are also present in the new authority's entry at or above a predetermined, configurable threshold. If so, the software can suggest that the newly entered authority is appropriate for the user with respect

to that taxonomy category and additionally allow the user to view the authority and create new summaries.

Therefore, a person of ordinary skill in the art would have known of numerous, equivalent data communications techniques, implementable in software, to receive notice and a data stream comprising keywords.

The Patent Examiner states that processing the user taxonomy table for keywords associated with one or more predetermined elements of the user taxonomy table for keywords associated with each of those predetermined elements of the user taxonomy table in claim 16 is unclear because the method of obtaining keywords from a table is unclear. (First Office Action, page 6) “Keywords” and their generation were discussed above. A table was a structural term of art readily understood by programmers in 2001 as was obtaining data from a table, e.g. using an SQL query or other database software. *Business Objects, S.A.*, 393 F.3d at 1368.

With respect to “examining the keywords associated with each of the predetermined elements of the user taxonomy table against the keywords received associated with the new raw data,” software, as is old in the art, may be used to analyze (examine) data files. (See, e.g., Exhibit A, Ullman, page 81 (discussing operations and interrogations))

At ¶ [0059], the ‘521 Application teaches

In an embodiment, the user may also set a threshold value for searches such as at 622 whereby a query containing a plurality of keywords for the search requires an authority or summary to **possess at least that threshold of keyword occurrences to be considered a positive or relevant result**. By way of example and not limitation, a user query may contain four keywords and have a threshold of fifty percent. Any summary or authority, depending on the search requested, that has at least two of the four keywords is then presented to the user as a positive or relevant search result. Others, e.g. with only one keyword, are not presented.

(emphasis added) Although it is improper to import extraneous limitations from the specification into the claims and Applicants are not doing so, claims are read in light of the specification<sup>7</sup> and this is but one example of how one of ordinary skill in these arts would have understood “relevance factor” in October 2001 and how to generate such a relevance factor. As is clear from the context, after analyzing two data sets, a mathematical component that reflects relevance of one element in a first data set to another in the second data set, e.g. a percentage of terms present, would have been readily apparent to anyone of ordinary skill in these arts.

At ¶ [0061], the ‘521 Application teaches

Additionally, when new raw data are added to authority table 22a (FIG. 2) or when new elements are added into user question/FAQ table 12c (FIG. 2), the generated keywords may be used to suggest one or more taxonomy categorizations to the user. **By way of example and not limitation, software executing at user computer 10,30,50 (FIG. 1) may receive notice that a new entry has been added to authority table 22a (FIG. 2). That software may gather the keywords associated with the new element of authority table 22a (FIG. 2) and then, using the user’s taxonomy table 12a (FIG. 2), gather keywords associated by the user with that user taxonomy table 12a (FIG. 2) such as on an element by element basis. For each element in the taxonomy, the software can determine if those keywords are also present in the new authority’s entry at or above a predetermined, configurable threshold. If so, the software can suggest that the newly entered authority is appropriate for the user with respect to that taxonomy category and additionally allow the user to view the authority and create new summaries.”**

(emphasis added).

Thus, claim 16 is fully, clearly, and concisely described. Accordingly, claim 16 satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner should be reversed.

#### f) Claim 17

“Query” was a well-known term of database art. *See, e.g., Business Objects, S.A.,*

---

<sup>7</sup> *Vitronics, Corp. v. Conception, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996).

393 F.3d at 1368; (Exhibit A, Ullman, page 81 (discussing “queries”)). Database systems/language often have query languages. *Id.* Formulating a query is a term of art meaning to create, craft, type in a query. *Id.* For example, ¶¶ [0057-0058] describe Fig. 6 which illustrates an entry form and teach that the queries may

contain query may designate certain words as required, such as by using a “+” symbol, and words to be omitted, such as by using a “-” symbol. Additionally, the user may use proximity search designators including “within sentence,” “within paragraph,” and/or “within ‘n’ words of ‘phrase’” designators. In advanced searches, a user may further be able to designate additional search conditions for one or more fields present in authority table 12a, e.g. dates, authors, type of authority, and the like.

In fact, even most widely used Internet search engines in 2001 used forms to let users formulate queries, e.g. [www.altavista.com](http://www.altavista.com) or [www.google.com](http://www.google.com).

In computer science in 2001, to filter means that “a program or routine ... blocks access to data that meet a particular criterion.” (Exhibit C, [www.dictionary.com](http://www.dictionary.com)) ‘521 Application ¶¶ [0057-0058] describe an embodiment of filtering. The ‘521 Application teaches that, for these embodiments,

the user may elect to designate how the keywords are to be viewed for a search to produce a result. ... In advanced searches, a user may further be able to designate additional search conditions for one or more fields present in authority table 12a, e.g. dates, authors, type of authority, and the like. ... [T]he user may also set a threshold value for searches such as at 622 whereby a query containing a plurality of keywords for the search requires an authority or summary to possess at least that threshold of keyword occurrences to be considered a positive or relevant result.

Moreover, it is readily apparent that, in context and as taught in the specification, summaries and/or authorities may be searched based on the keywords and the results would obviously be those summaries and/or authorities that meet the user’s filtering criteria, i.e. the query.

Thus, claim 17 is fully, clearly, and concisely described. Accordingly, claim 17

satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner reversed.

**g) Claim 18**

Claim 17, from which claim 18 depends, incorporates a system of claim 1 which comprises numerous tables including a searchable authority table, user definable taxonomy table, and searchable summary table.<sup>8</sup> These tables provide the antecedent basis to the term “the tables” in claim 18. However, the ‘521 Application teaches that users may have access to other sources of authorities. For example, in ¶ [0082], the ‘521 Application teaches

By way of further example and not limitation, the ‘521 Application may be used in conjunction with other free or for-fee services, such as those available through the Internet 100, such as by being a front end to those other services. **Acting as an intermediary, the ‘521 Application allow users to contain and refine knowledge and references uncovered during prior searches and integrate new information into the user’s captured knowledge and references.** By way of example and not limitation, in this manner a user may continue to use services such as LEXIS/NEXIS™ offered by REID ELSEVIER, INC. or WESTLAW™ offered by THE THOMSON GROUP but not be required to utilize the taxonomies used by those services. Similarly, users may access search engines available through the Internet 100 such as YAHOO™ or GOOGLE™ which either have their own taxonomy or offer no taxonomy, and integrate information uncovered with those search into the user’s captured knowledge and references.

(emphasis added) Therefore, the ‘521 Application teaches that either a user or the system itself via software can access external sources.

These steps involve allowing the user access from within the claimed invention’s system, e.g. using a form, to other sources/tables, e.g. outside, search engines. This would have been readily understood by anyone of ordinary skill in these arts.

Thus, claim 18 is fully, clearly, and concisely described. Accordingly, claim 18

---

<sup>8</sup> Note: it is the taxonomy data that are user definable, not the taxonomy table, i.e. the data themselves and not the table’s structural layout.

satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner reversed.

**h) Claim 19**

As noted above, to filter means that “a program or routine ... blocks access to data that meet a particular criterion” and the ‘521 Application, e.g. ¶¶ [0057-0058], describes an embodiment of filtering. In its described embodiments, that which the ‘521 Application teaches as filtering is one form of limiting searching.

Thus, claim 19 is fully, clearly, and concisely described. Accordingly, claim 19 satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner reversed.

**i) Claim 20**

Forms are illustrated in the figures, e.g. Fig. 4, illustrating viewing such as viewing of raw data at Fig. 4 (*see* button marked “FULL TEXT” in Fig. 4) or summary data at 420. Paragraph [0053] teaches:

By way of example and not limitation, the user may retrieve the previously entered element from authority table 12a, view the raw data associated with that element, highlight or mark a section of the raw data to be captured, and signal a desire to capture the highlighted or marked section as a summary. **This may be accomplished, by way of example and not limitation, by using a mouse to highlight the section and then using a right-button click of the mouse to bring up a menu that has an entry to allow capture.**

(emphasis added) The claimed method is not limited to buttons, however, as use of buttons is but a single method of allowing a user to elect a view of data.

Thus, claim 20 is fully, clearly, and concisely described. Accordingly, claim 20 satisfies 35 U.S.C. §112 ¶ 1 and the Patent Examiner should be reversed.

**B. Rejections under 35 U.S.C. §112 ¶ 2**

In the First Office Action, the Patent Examiner rejected claim 15 as being indefinite because it recites “the description of the raw data comprises at least one of the

raw data, a pointer to the raw data, a description of a file containing the raw data, and a description of a remote source location of a file containing the raw data.” The Patent Examiner could not understand how raw data could be a description for itself. By way of example, a formula could be raw data (e.g.,  $A = B + C$ ) and its best description could simply be that formula. The Applicants responded accordingly and the Patent Examiner appears to have conceded this argument to Applicants in the Final Office Action.

**C. Claims 1 and 4-11 Are Not Anticipated By Chakrabarti**

“Anticipation under 35 U.S.C. § 102 means lack of novelty, and is a question of fact. To anticipate, every element and limitation of the claimed invention must be found in a single prior art reference, arranged as in the claim.” *Karsten Mfg. Corp. v. Cleveland Golf Co.*, 242 F.3d 1376, 1383 (Fed. Cir. 2001) (emphasis added); MPEP 2131. Chakrabarti does not disclose every element and limitation of claim 1 arranged as in Claim 1 and the Patent Examiner’s rejection was incorrect.

First, Charabarti does not disclose an authority table comprising “raw” data. “Table” is as defined and discussed above. *See, e.g., Business Objects, S.A.*, 393 F.3d at 1368. As used in the ‘521 Application, the authority table comprises “raw data,” i.e. text data, graphics data, audio data, video data, documents, images, and/or references to the actual data, such as a URL or other file link. (‘521 Application ¶¶ [0023, 0027]) Charabarti does not disclose any such authority table comprising raw data. Instead, Chakrabarti discloses that its “database 18 can include **plural** tables 22 **that in turn include information related to Web documents** (emphasis added).” (‘224 Patent, Col. 3, ll. 36-39) Information related to a Web document is not the actual data in that document at all, i.e. its contents, but only something amorphously related (and undefined

in Chakrabarti) to that document and is therefore not an “authority,” as used in the ‘521 Application and as claimed in claim 1. Moreover, Chakrabarti is silent as to what his “information related to” actually is and gives less information on relations between his plural tables 22 than is present in the ‘521 Application. Further, a Web document is not a table. Therefore, Chakrabarti does not disclose any table comprising “raw data.”

Additionally, the Patent Examiner is wrong when he asserts that software is “inherent” in Chakrabarti’s “web site Fig 1, 14” to provide access to and management of the authority table. (First Office Action, page 10; Final Office Action, page 4) Chakrabarti’s plural tables 22 are local to computer 12 and are not disclosed as being managed by anything outside that local computer 12, i.e. a server located via the Web 14. Thus, only the local computer 12 manages all tables in Chakrabarti. (‘224 Patent, Col. 3, ll. 32-38 (“Also, **the computer 12 accesses a database 18 via a data path 20**, it being understood that the data path 20 can be established by an internal computer bus, LAN, WAN, or other communication path. In any case, the **database 18 can include plural tables 22** that in turn include information related to Web documents, as indicated by the data path 24.”)) Therefore, Chakrabarti does not disclose a server comprising a searchable authority table.

Second, the Patent Examiner’s citation to Chakrabarti’s Fig. 2 step 36 is incorrect as it does not disclose a user manipulatable taxonomy table. (First Office Action, page 10; Final Office Action, page 4) Chakrabarti’s taxonomy table is not user manipulatable. Importantly, Chakrabarti teaches away from the user manipulating the taxonomy table, i.e. manipulating its data, by teaching that his computer software alone does the classification of documents. (See, e.g., Chakrabarti Col. 4, ll. 54-55; Col. 6, ll. 14-17 (“In



addition to the above optimization in which a left outer table join is used to classify a document, a bulk probe can also be used as indicated at block 40 of FIG. 2 in which a 'best' path is taken down the taxonomy tree.)) In the applicable claims of the '521 Application, the user creates the data in the taxonomy table, e.g. via a form such as illustrated at Fig. 3, and uses those data when the user classifies the authorities and their summaries. Chakrabarti further teaches away from user involvement by teaching that "the invention is a general purpose computer programmed according to the inventive steps herein to classify documents in a taxonomy." (Chakrabarti '224 Patent, Col. 2, ll. 2-4) Thus, documents are automatically classified in Chakrabarti by his software. ('224 Patent, Col. 2, ll. 63-66) Accordingly, Chakrabarti does not disclose a user computer comprising at least one user definable taxonomy table manipulatable by a user with adequate access permission to manipulate the user taxonomy table.

Third, the Patent Examiner's citation to Chakrabarti's Fig. 2 step 36 and Col. 4 ll. 54-56 is incorrect as it does not disclose a user manipulatable summary table. (First Office Action, page 10; Final Office Action, page 4) In fact, Chakrabarti does not disclose any summary table and the word "summary" does not even appear in Chakrabarti. All that Chakrabarti discloses is that documents sought to be classified can be referred to as "test" documents and can be represented in one or more tables. (Chakrabarti '224 Patent, Col. 4, ll. 41-43) These "representations" are not summaries, much less user created summaries, as claimed in the '521 Application.

In addition, in the applicable claims of the '521 Application each element in a user summary table (i.e., the user summary table's record containing user created summary data) must be linked to at least one element (a category) in a user taxonomy

table as well as to a single element (an authority) in the authority table. The claimed summary data are manageable by a user. Chakrabarti has no user managed summary data and does not link anything to an authority table. Thus, Chakrabarti does not disclose a user computer comprising at least one searchable summary table that is related to the authority and user taxonomy tables and which is manageable by user with adequate access permission to manage the summary table.

Fourth, Chakrabarti fails to disclose any software that allows a user to manipulate the tables, i.e. the data in the tables, especially the taxonomy table or the summary table. In fact, Chakrabarti is completely devoid of any teaching that users have an ability to manipulate (or even perceive) the tables or the data within those tables. Importantly, Chakrabarti teaches away from user involvement by teaching that “supervised learning” (i.e., computer automated processing) can be used, “wherein a few training documents initially are assigned to the various nodes of a taxonomy and subsequent documents are then classified [by his disclosed software] based on comparisons with the training documents.” (Chakrabarti ‘224 Patent, Col. 1, ll. 26-31)

Accordingly, Chakrabarti fails to disclose each and every element of Claim 1, arranged as in Claim 1. Thus, Applicants respectfully request that the Patent Examiner’s rejection of claim 1 under 35 U.S.C. §102 be reversed and Claim 1 allowed.

With respect to Claim 4, Claim 4 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 4 depends from an otherwise allowable claim, Claim 4 is itself allowable. Thus, Applicants respectfully request that the Patent Examiner’s rejection of claim 4 under 35 U.S.C. §102 be reversed and Claim 4 allowed.

With respect to Claim 5, Claim 5 depends from Claim 1 which is distinguishable

over Chakrabarti. Because Claim 5 depends from an otherwise allowable claim, Claim 5 is itself allowable. Moreover, Chakrabarti never mentions legal data, medical data, educational data, manufacturing data, scientific data, or repair data. Importantly, Chakrabarti's Web "raw data" files are not included in his "plural tables 22." Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 5 under 35 U.S.C. §102 be reversed and Claim 5 allowed.

With respect to Claim 6, Claim 6 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 6 depends from an otherwise allowable claim, Claim 6 is itself allowable. Further, as discussed above, Chakrabarti does not disclose an authority table and there is no callout 18 in Chakrabarti's Fig. 1. (First Office Action, page 10; Final Office Action, page 4) Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 6 under 35 U.S.C. §102 be reversed and Claim 6 allowed.

With respect to Claim 7, Claim 7 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 7 depends from an otherwise allowable claim, Claim 7 is itself allowable. As noted above, the server (Chakrabarti's internet web server) does not have nor allow access to any of Chakrabarti's plural tables 22, which are local to his computer 12. Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 7 under 35 U.S.C. §102 be reversed and Claim 7 allowed.

With respect to Claim 8, Claim 8 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 8 depends from an otherwise allowable claim, Claim 8 is itself allowable. Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 8 under 35 U.S.C. §102 be reversed and Claim 8 allowed.

With respect to Claim 9, Claim 9 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 9 depends from an otherwise allowable claim, Claim 9 is itself allowable. Moreover, Chakrabarti does not disclose that user computer 12 uses Internet browsing software executable at the user computer to access the predetermined portions of database 18. Instead, all of database 18 is accessed via non-Internet computer 12. Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 9 under 35 U.S.C. §102 be reversed and Claim 9 allowed.

With respect to Claim 10, Claim 10 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 10 depends from an otherwise allowable claim, Claim 10 is itself allowable. Moreover, there is no disclosure of a user being allowed to submit a query to the query software via the Internet. The only query mentioned by Chakrabarti is "The first query at the root node is identical to the query preceding the left outer join shown above. However, rather than completely finish taxonomy traversal before classifying another document, the present invention processes entire taxonomy nodes at a time with plural documents." (Chakrabarti '224 Patent, Col. 6, ll. 36-39) This "query" is the result of his software executing at computer 12, i.e. an SQL command, and most definitely not a user submitting a query, i.e. a question, via the Internet. Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 10 under 35 U.S.C. §102 be reversed and Claim 10 allowed.

With respect to Claim 11, Claim 11 depends from Claim 1 which is distinguishable over Chakrabarti. Because Claim 11 depends from an otherwise allowable claim, Claim 11 is itself allowable. Moreover, there is simply no disclosure within Chakrabarti of any type of user interface. Chakrabarti does not disclose users

interfacing to his system or using his methods, all of which are processed without human intervention. Thus, Applicants respectfully request that the Patent Examiner's rejection of claim 11 under 35 U.S.C. §102 be reversed and Claim 11 allowed.

Accordingly, Chakrabarti does not function as an anticipatory reference under 35 U.S.C. §102(e) of claims 1 and 4-11 and these claims should be allowed.

**D. Claim 2 Is Not Rendered Obvious By Chakrabarti In View Of Price**

Obviousness is ultimately a conclusion of law based on underlying findings of fact. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). These underlying factual findings include: (1) the scope and content of the prior art; (2) the level of ordinary skill in the art; (3) the differences between the claimed invention and the prior art; and (4) the extent of any proffered objective indicia of non-obviousness. *Id.* at 17-18. When patentability turns on the question of obviousness, the search for and analysis of the prior art includes evidence relevant to the finding of whether there is a teaching, motivation, or suggestion to select and combine the references relied on as evidence of obviousness. *McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 1351-52 (Fed. Cir. 2001) ("the central question is whether there is reason to combine [the] references," a question of fact drawing on the Graham factors). "The factual inquiry whether to combine references must be thorough and searching." *Id.*; *Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1124-25 (Fed. Cir. 2000) ("a showing of a suggestion, teaching, or motivation to combine the prior art references is an 'essential component of an obviousness holding'") (quoting *C.R. Bard, Inc., v. M3 Systems, Inc.*, 157 F.3d 1340, 1352 (Fed. Cir. 1998)); *In re Dembiczak*, 175 F.3d 994, 999 (Fed. Cir. 1999) ("Our case law makes clear that the best defense against the subtle but powerful attraction of a hindsight-based obviousness

analysis is rigorous application of the requirement for a showing of the teaching or motivation to combine prior art references.”); *In re Dance*, 160 F.3d 1339, 1343 (Fed. Cir. 1998) (there must be some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the applicant); *In re Fine*, 837 F.2d 1071, 1075 (Fed. Cir. 1988) (teachings of references can be combined only if there is some suggestion or incentive to do so.)

It is fundamental that the Patent Examiner must show where all of the claimed limitations may be found in the prior art to establish even a *prima facie* case of obviousness. A Patent Examiner’s rejection must be based on objective evidence and cannot be based on subjective belief or conclusory statements. *In re Lee*, 277 F.3d 1338, 1343 (Fed. Cir 2002).

With respect to Claim 2, which depends from Claim 1, as discussed above Chakrabarti does not disclose that which is claimed in Claim 1. Moreover, there is simply no reason to combine Chakrabarti with Price and no valid suggestion, motivation, nor teaching to do so.

Chakrabarti discloses a system and method for optimizing I/O to low-level index access during bulk-routing through a taxonomy to classify documents, e.g., Web pages, in the taxonomy, for use in high-dimensional similarity searches. (Chakrabarti, Col. 1, ll. 14-17) On the other hand, Price (the “‘691 Patent”) discloses a method and apparatus for routing calls in a switched digital network and a processor for use in such a network are provided in which tables of call control information store information for different categories of call control information in separate parts of the table or in separate tables. (Price (‘691 Patent), Abstract)

There is simply no motivation, suggestion, or teaching that would even hint at combining these references:

- Chakrabarti is looking to help aid high-dimensional searching in multimedia databases that contain text documents, audio, and video and never mentions avoidance of duplication.
- Price is directed towards avoidance of duplication of information across a call control table as well as portability of tables and never mentions whose “searching” is limited to examining a call control table for call control information for a call type.
- Chakrabarti is in the field of document access control.
- Chakrabarti never discloses any use for avoidance of duplication.
- Price is in the field of telephony.
- Price never discloses use with anything other than telephony data.

Applicants respectfully submit that there is no motivation, suggestion, or teaching of the desirability of making this specific combination suggested by the Patent Examiner. (First Office Action, page 13; Final Office Action, page 7) The Patent Examiner’s reasoning that a person of ordinary skill would be motivated to combine high-dimensional searching in multimedia databases to provide a call control table for storing call control information defies logic and is simply falling prey to hindsight reasoning. *Atlas Powder Company v. E.J. du Pont de Nemours*, 750 F.2d 1569, 1574 (Fed. Cir. 1984).

Even so, combining these two references – and there is no reason, suggestion, or motivation as to why anyone would – would not result in the invention claimed in Claim 2. Even the combination of these two does not disclose a system for organization of information, comprising (a) a server comprising a searchable authority table comprising authorities (raw data) or (b) a user computer further comprising at least one user

definable, user manipulatable taxonomy table, (c) at least one user manageable searchable summary table that is related to the authority table and the user taxonomy table, and (d) software executable in the user computer to provide access to and manipulation of the taxonomy table and the summary table.

Accordingly, the Patent Examiner should be reversed and claim 2 allowed.

**E. Claim 3 Is Not Rendered Obvious By Chakrabarti Alone invalid**

As discussed above, the Patent Examiner is apparently confused about the summary table. In the Final Office Action at pages 9-10, the Patent Examiner states that the specification is unclear because the purpose of the summary table is confusing.

Claim 3 and the '521 Application are both clear on the purpose of the summary table. The summary table does not contain stored results obtained from a data gathering step. (Final Office Action page 8) Instead, the summary table comprises user created data. ('521 Application, Claim 1(c)(ii)(2); Fig. 4) The '521 Application, in the following paragraphs, teaches the following:

[0023] ("**database 22 may comprise user manipulatable data, referred to herein as "summary" data, typically in summary table 12a**");

[0044] ("User summary table 12b may be part of user portion 300 and **contains an element for each item of information the user desires to link to at least one element in user taxonomy table 12a**. In the currently preferred embodiment, each element in user summary table 12b, sometimes referred to herein as a "summary," must be linked to at least one element in user taxonomy 12a as well as to a single element in authority table 22a in backend portion 200.");

[0044] ("One currently envisioned method of relating user summary table 12b to authority table 22a is to use an intermediate table such as user knowledge table 12e which relates one element in authority table 22a to one element in user summary table 12b, although one-to-many relationships may be defined in this or other manners as well.");

[0046] ("summaries captured such as in user summary table 12b may be



used in conjunction with user question/FAQ table 12c and taxonomy table 12a to create a user-based cylopedia or reference work.”);

[0053] (“By way of example and not limitation, the user may retrieve the previously entered element from authority table 12a, view the raw data associated with that element, highlight or mark a section of the raw data to be captured, and signal a desire to capture the highlighted or marked section as a summary. This may be accomplished, by way of example and not limitation, by using a mouse to highlight the section and then using a right-button click of the mouse to bring up a menu that has an entry to allow capture. In this manner, for example, a capture function could additionally capture the reference parameters such as word offset information for captured text data. The user may additionally associate the captured summary with one or more elements in taxonomy table 12a. Optionally, the user may create new elements in user taxonomy table 12a for use with the summary. The user may additionally edit the summary data captured and/or replace or augment all or a portion of the summary with whatever the user wishes, by way of example and not limitation including audio augmentation, video augmentation, and the like.”);

[0055] (“Accordingly, the user can synopsise or summarize the raw data into information relevant to that user. In an additionally envisioned embodiment, an automatic summary option may be present to synopsise or suggest a synopsis of an authority or a portion of an authority. Further, a user may use form 400 to search for authorities and their summaries.”); and

Fig. 4, which shows an entry screen form in which a user creates in a summary in dialog box 420 (see discussion of [0055] above)

As can be seen from these references from the ‘521 Application, the summary table contains records containing data created by a user in which the user summarizes/synopsizes what the user thinks important regarding that authority. That user-created summary record is then linked to the authority that the user was summarizing, e.g. by virtue of being part of linked tables in one or more databases. (See, e.g., Fig. 4 showing a summary (at 420) linked to an authority (*U.S. Fish and Wildlife Serv. v. Sierra Club*)) The summary table itself summarizes nothing: it contains summarization records which the user created to be summaries having meaning to the

user.

Claim 3 depends from Claim 1. Chakrabarti, as discussed herein above, does not disclose the limitations of Claim 1 and never even so much as hints at summaries. Because the summary table is a user manipulatable table, management of the summarization table that comprises creating, modifying, and deleting elements of the summary table and associating elements of the summary table with at least one element of the user taxonomy table would be by the user. ('521 Application Claim 1(b)(ii)(c); Exhibit A, Ullman, page 81) As discussed above, Chakrabarti does not even suggest user manipulation and teaches away from user manipulation by teaching an automated system.

Accordingly, the Patent Examiner should be reversed and Claim 3 allowed.

**F. Claims 12-20 Should Be Allowed**

Claims 12-16 and 17-20 claim methods of using the system of Claim 1. As discussed herein, Claims 12 and 17 are fully enabled, not anticipated by the cited references, and not rendered obvious by those references. Accordingly, claim 12 and its dependent claims 13-16 and claim 17 and its dependent claims 18-20 should be allowed.

**VIII. Claims Appendix**

The following are the claims of the '521 Application as they stand on appeal.

1. A system for organization of information, comprising:
  - a. a server comprising a searchable authority table;
  - b. a data communications device operatively in communication with the server;
  - c. a user computer operatively in communication with the data communications device, the user computer having access to the searchable

authority table, the user computer further comprising:

- i. at least one user definable taxonomy table, wherein the taxonomy table:
  - (1) is accessible to the user computer; and
  - (2) is manipulatable by a user with adequate access permission to manipulate the user taxonomy table; and
- ii. at least one searchable summary table, wherein the summary table:
  - (1) is accessible to the user computer;
  - (2) is related to the authority table and the user taxonomy table; and
  - (3) is manageable by a user with adequate access permission to manage the summary table;
- d. software executable in the server to provide access to and management of the authority table; and
- e. software executable in the user computer to provide access to and manipulation of the taxonomy table and the summary table.

2. The system of claim 1 wherein manipulation of the taxonomy table comprises creating, modifying, associating elements in the taxonomy table with other elements in the taxonomy table for dynamic alternate presentation, rearranging, and deleting elements of the user taxonomy table.

3. The system of claim 1 wherein management of the summarization table comprises

creating, modifying, and deleting elements of the summary table and associating elements of the summary table with at least one element of the user taxonomy table.

4. The system of claim 1 wherein the user computer has access to a plurality of searchable raw data via the data communications device.

5. The system of claim 1 wherein the authority table comprises descriptions of at least one of legal data, medical data, educational data, manufacturing data, scientific data, repair data, audiovisual data, and entertainment data.

6. The system of claim 1 wherein the server further comprises a database comprising the authority table.

7. The system of claim 1 further comprising:

- a. a data network accessible to the server and the user computer via the data communications device; and
- b. query software executing at least partially in the server;
- c. whereby computers with access to the data network may be allowed to access predetermined portions of the authority table.

8. The system of claim 7 wherein the authority table comprises descriptions of data located via the Internet.

9. The system of claim 7 wherein:
- a. the data network comprises the Internet;
  - b. the user computer uses Internet browsing software executable at the user computer to access the predetermined portions of the database; and
  - c. the server is an Internet service provider;
  - d. wherein the user computer further has access to a plurality of searchable raw data via the Internet.
10. The system of claim 9, wherein a user is allowed to submit a query to the query software via the Internet.
11. The system of claim 1 further comprising a user interface to the system comprising at least one of an HTML user interface, a non-database language user interface, and a database language interface.
12. A method of creating summarized information for later access for a system of claim 1, comprising:
- a. capturing raw data into an programmatically manipulatable format;
  - b. storing the programmatically manipulatable format of the raw data into the authority table;
  - c. capturing a description of a source of the raw data;
  - d. storing the description into the authority table while associating the description with the data in the authority table;

- e. programmatically retrieving the raw data from the authority table by a user;
- f. allowing the user to summarize at least a portion of the retrieved raw data, the summarization to be stored in the summary table;
- g. examining one or more elements in the user taxonomy table for an appropriate taxonomy table element to associate with the summarization;
- h. creating a new element in the user taxonomy table describing an appropriate taxonomy element to associate with the summarization if an appropriate element is not already present in the taxonomy table;
- i. associating the summarization with the appropriate element of the taxonomy table;
- j. associating the summarization with the authority table; and
- k. storing the summarization and associations in the summary table.

13. The method of claim 12 wherein step (a) further comprises:

- i. parsing the raw data prior to storing the captured raw data in the authority table;
- ii. generating keywords from the parsed raw data;
- iii. saving the keywords in a keyword table associated with the authority table; and
- iv. storing the raw data description in the authority table.

14. The method of claim 12 wherein step (k) further comprises:

- i. parsing the summarization prior to storing the summarization in the summary table;
- ii. generating keywords from the parsed summarization;
- iii. saving the keywords in a keyword table associated with the summary table; and
- iv. storing the summarization in the summary table.

15. The method of claim 12 wherein the description of the raw data comprises at least one of the raw data, a pointer to the raw data, a description of a file containing the raw data, and a description of a remote source location of a file containing the raw data.

16. The method of claim 12 further comprising:

- a. receiving a notice of addition of new raw data to the authority table at the user computer;
- b. receiving keywords associated with the new raw data at the user computer;
- c. processing the user taxonomy table for keywords associated with one or more predetermined elements of the user taxonomy table for keywords associated with each of those predetermined elements of the user taxonomy table;
- d. examining the keywords associated with each of the predetermined elements of the user taxonomy table against the keywords received associated with the new raw data;
- e. generating a relevance factor for the new raw data based on the

examination; and

- f. suggesting the new raw data to the user as relevant for each of the predetermined elements of the user taxonomy table where the relevance factor is at a predetermined threshold level in each of the predetermined elements of the user taxonomy table.

17. A method of searching a system of claim 1 for relevant information, comprising:

- a. formulating a query at a user workstation;
- b. analyzing the query for keywords;
- c. obtaining user filtering input for tables to be searched;
- d. searching for keywords against the tables using the user filtering input;  
and
- e. returning search results to the user.

18. The method of claim 17, further comprising at least one of:

- a. allowing the user to continue the search outside the tables when the number of search results occurs below a predetermined threshold; and
- b. allowing the user to continue the search outside the tables on a user initiated command.

19. The method of claim 17, wherein the filtering input comprises at least one of limiting searches to a selected element of the taxonomy, limiting searches to a plurality of selected elements of the taxonomy, limiting searches to all elements of the taxonomy,



limiting searches based on fields present for an authority table element, and limiting searches based on fields present for a summary table element.

20. The method of claim 17 wherein a user viewing a summary table element may be allowed to view the raw data from which that summary was derived, the allowing comprising at least one of selecting a region on a display at the user computer in which the summary is being displayed, selecting a command button on the display, and using one or more keys on a keyboard associated with the user computer.

**IX. Evidence Appendix.**

Appendix IX contains copies of evidence being submitted pursuant to §§ 1.130, 1.131, or 1.132. The Patent Examiner has not yet entered any evidence and therefore none has been relied upon by Appellants in this appeal, other than the Office Actions, their associated prior art, and notices in the file history.

This Appendix includes Exhibits A, B, and C as referred to in the Appeal Brief.

**X. Related Proceedings Appendix**

There are no decisions rendered by a court or the Board in any related proceeding.

Respectfully submitted,

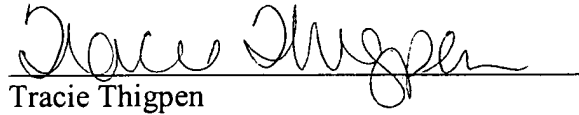
February 3, 2006



Gary R. Maze  
Reg. No. 42,851  
Duane Morris LLP  
3200 Southwest Freeway, Suite 3150  
Houston, TX 77027  
Phone: 713.402.3900  
Fax: 713.402.3901

# **CERTIFICATE OF MAILING 37 CFR 1.10**

I hereby certify that this correspondence along with any referred to as attached or enclosed is being deposited with the United States Postal Service as Express Mail, Label No. EV572318300, postage prepaid in an envelope addressed to: Mail Stop Appeal Brief Patents, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on February 3, 2006.

  
Tracie Thigpen

# Principles of Database Systems

Jeffrey D. Ullman  
Stanford University

COMPUTER SCIENCE PRESS

## COMPUTER SOFTWARE ENGINEERING SERIES

ELLIS HOROWITZ, Editor  
*University of Southern California*

CALINGAERT  
*Assemblers, Compilers, and Program Translation*

CARBERRY, KHALIL, LEATHRUM, and LEVY  
*Foundations of Computer Science*

EVEN  
*Graph Algorithms*

FINDLAY and WATT  
*PASCAL: An Introduction to Methodical Programming*

HOROWITZ and SAHNI  
*Fundamentals of Computer Algorithms*

HOROWITZ and SAHNI  
*Fundamentals of Data Structures*

ULLMAN  
*Principles of Database Systems*

# Preface

Copyright © 1980 Computer Science Press, Inc.

Printed in the United States of America

All rights reserved. No part of this work may be reproduced, transmitted, or stored in any forms or by any means, without the prior written consent of the Publisher.

*Computer Science Press, Inc.  
9125 Fall River Lane  
Potomac, Maryland 20854*

1 2 3 4 5 6 85 84 83 82 81 80

## Library of Congress Cataloging in Publication Data

Ullman, Jeffrey D. 1942-  
Principles of database systems.

Bibliography: p.  
Includes index.

1. Data base management. I. Title.  
QA76.9.D3U44 001.6'4 79-20071  
US ISBN 0-914894-13-7  
UK ISBN 0-273-08476-3

It is evident that a course in database systems now plays a central role in the undergraduate and graduate programs in computer science. However, unlike the more traditional and better established systems areas, like compilers and operating systems, where a good mix of principles and practice was established many years ago, the subject matter in database systems has been largely descriptive.

This book is developed from notes I used in a course at Princeton that attempted to bring database systems into the mainstream of computer science. The course was taught to a mix of seniors and first-year graduate students. In it, I tried to relate database ideas to concepts from other areas, such as programming languages, algorithms, and data structures. A substantial amount of descriptive material was included, since students, being used to conventional programming languages, may find query languages rather unusual. The data structures relevant to databases are also somewhat different from the kinds of structures used in conventional programming, since the large scale of a database makes practical many structures that would be only of theoretical interest otherwise.

However, I added to the mix of topics the relevant theory that is now available. The principal concepts that have been found useful are concerned with relations and with concurrency. I have devoted a large portion of the book to a description of relations, their algebra and calculus, and to the query languages that have been designed using these concepts. Also included is an explanation of how the theory of relational databases can be used to design good systems, and a description of the optimization of queries in relation-based query languages. A chapter is also devoted to the recently developed protocols for guaranteeing consistency in databases that are operated on by many processes concurrently.

## Exercises

Each chapter includes exercises to test basic concepts and, in some cases, to extend the ideas of the chapter. The most difficult exercises are marked with a double star, while problems of intermediate difficulty have a single star.

[1978]. Hashing techniques are surveyed in Morris [1968], Knuth [1973], and Maurer and Lewis [1975]. The selection of a physical database scheme from among alternatives is discussed by Gottlieb and Tompa [1973].

Fast ordered list searching is discussed by Yao and Yao [1976]. The  $\log \log n$  complexity of interpolation search is shown there; see also Perl, Itai and Avni [1978]. The B-tree is from Bayer and McCreight [1972], where it was presented as a dense index, as described in Section 2.6, rather than in the general form of Section 2.5. The articles by Held and Stonebraker [1978] and Snyder [1978] contain an interesting discussion of the merits of their use in database systems. Yao [1978] analyzes the expected occupancy of 2-3 trees (B-trees with two or three children per node) and shows that on the average, interior nodes have somewhere between 21% and 30% waste space.

The choice of secondary indices is discussed by many people, including Lum and Ling [1970] and Shkolnick [1975]. The point of view generally taken is that the contents of the database can be known before indices are selected. Comer [1978] shows optimal selection NP-complete. (See Aho, Hopcroft, and Ullman [1974] or Garey and Johnson [1979] for a discussion of how NP-completeness implies a problem cannot be solved efficiently.)

The use of partitioned hashing functions for partial match retrieval was considered in its generality by Rivest [1976], and the design of such hashing functions was also investigated by Burkhard [1976]. Theorem 2.1 on the shape of optimal hashing functions is by Bolour [1979]; Theorem 2.2, the case where queries specify one field, is from Rothnie and Lozano [1974], and Theorem 2.3 is from Bolour [1979] and Aho and Ullman [1979]. A related idea, called superimposed codes, also has applications to database systems, as described in Roberts [1978], for example.

# 3

## The Three Great Data Models

In Chapter 1 we mentioned the role of a data model as a basis for data definition and manipulation languages. Now let us consider the three most important such models — the relational, network, and hierarchical. In subsequent chapters we shall explore the relational model predominantly, for reasons we shall discuss at the end of the chapter, after we have had a chance to see the three models in detail. Briefly, the relational model provides the descriptive power of the other models with fewer distinct concepts to learn or special cases to handle. Chapters 4–6 will discuss aspects of the relational model in detail, including a number of high-level data manipulation languages that have been developed for this model.

However, the present state of the art is such that existing commercial systems are almost exclusively based on one of the other two models, a situation that is expected to change slowly. We shall in later chapters consider the hierarchical and network models in some detail too, covering the dominant system designs in these two categories, IBM's hierarchical IMS (Information Management System) and the DBTG (data base task group) network proposal.

### 3.1 The Relational Data Model

The mathematical concept underlying the relational model is the set-theoretic *relation*, which is a subset of the Cartesian product of a list of domains. A *domain* is simply a set of values. For example, the set of integers is a domain. So are the set of character strings, the set of character strings of length 20, the real numbers, the set  $\{0, 1\}$ , and so on. The *Cartesian product* of domains  $D_1, D_2, \dots, D_k$ , written  $D_1 \times D_2 \times \dots \times D_k$ , is the set of all  $k$ -tuples  $(v_1, v_2, \dots, v_k)$  such that  $v_1$  is in  $D_1$ ,  $v_2$  is in  $D_2$ , and so on. For example, if  $k=2$ ,  $D_1 = \{0, 1\}$ , and  $D_2 = \{a, b, c\}$ , then  $D_1 \times D_2$  is  $\{(0, a), (0, b), (0, c), (1, a), (1, b), (1, c)\}$ .

A *relation* is any subset of the Cartesian product of one or more domains. As far as databases are concerned, it is pointless to discuss infinite relations, so we shall assume that a relation is finite unless we state

otherwise. For example,  $\{(0,a), (0,c), (1,b)\}$  is a relation, a subset of  $D_1 \times D_2$  defined above. The empty set is another example of a relation.

The members of a relation are called *tuples*. Each relation that is a subset of  $D_1 \times D_2 \times \dots \times D_k$  is said to have *arity*  $k$ . A tuple  $(v_1, v_2, \dots, v_k)$  has  $k$  components; the  $i$ th component is  $v_i$ . Often we use the shorthand  $v_1 v_2 \dots v_k$  to denote the tuple  $(v_1, v_2, \dots, v_k)$ .

It helps to view a relation as a table, where each row is a tuple and each column corresponds to one component. The columns are often given names, called *attributes*.

*Example 3.1:* In Fig. 3.1 we see a relation whose attributes are CITY, STATE, and POP. The arity of the relation is three. For example,

(Miami, Oklahoma, 13880)

is a tuple.  $\square$

CITY	STATE	POP
San Diego	Texas	4490
Miami	Oklahoma	13880
Pittsburg	Iowa	509

Fig. 3.1. A relation.

### An Alternative Formulation of Relations

If we attach attribute names to columns of a relation, then the order of the columns becomes unimportant. In mathematical terms we view tuples as mappings from attributes' names to values in the domains of the attributes. This change in viewpoint makes certain relations equal that were not equal under the more traditional definition of a relation.

*Example 3.2:* Figure 3.2 shows two versions of the same relation in the set-of-mappings point of view. For example, as a mapping  $f$ , the tuple (Buffalo, W. Va., 831) is defined by  $f(\text{CITY}) = \text{Buffalo}$ ,  $f(\text{STATE}) = \text{W. Va.}$ , and  $f(\text{POP}) = 831$ . Note that the order in which the tuples are listed makes no difference in either viewpoint. However, in the traditional view of a tuple as a list of values, the tuples (Buffalo, W. Va., 831) and (W. Va., 831, Buffalo) would not be the same, and the two relations of Fig. 3.2 would not be considered the same.  $\square$

As existing relational systems allow the printing of columns of a relation in any order, we shall take the set-of-mappings definition of relations as the standard one. However, there are situations, such as when we discuss relational algebra in Section 4.1, where we shall want to use the set-of-lists definition for relations. Fortunately, there is an obvious method of converting between the two viewpoints. Given a relation in the set-of-lists

CITY	STATE	POP	STATE	POP	CITY
Buffalo	W. Va.	831	Utah	1608	Providence
Providence	Utah	1608	W. Va.	831	Buffalo
Las Vegas	N. M.	13865	N.M.	13865	Las Vegas

Fig. 3.2. Two presentations of the same relations.

sense, we can give arbitrary attribute names to its columns, whereupon it can be viewed as a set of mappings. Conversely, given a relation in the set-of-mappings sense, we can fix an order for the attributes and convert it to a set of lists.

### Relation Schemes

The list of attribute names for a relation is called the *relation scheme*. If we name a relation REL, and its relation scheme has attributes  $A_1, A_2, \dots, A_k$ , we often write the relation scheme as  $\text{REL}(A_1, A_2, \dots, A_k)$ . We should observe the analogy between a relation scheme and a record format as well as similar analogies between a relation and a file and between a tuple and a record. The difference in each case is one of the level of abstraction. That is, one possible implementation, among others, for a relation is as a file of records with a record format equal to the list of attributes in the relation scheme, and with one record for each tuple. However, one can conceive of many other implementations for a relation, and some of these will be discussed later in this section.

### Representing Data in the Relational Model

The collection of relation schemes used to represent information is called a (*relational*) *database scheme*, and the current values of the corresponding relations is called the (*relational*) *database*. We are, of course, free to create relations with any set of attributes for a relation scheme, and we can place any interpretation we wish on tuples. However, we can observe the typical pattern of usage if we recall one discussion of the entity-relationship model from Section 1.4. The data of an entity-relationship diagram is represented by two sorts of relations.

1. An entity set can be represented by a relation whose relation scheme consists of all the attributes of the entity set. If this entity set is one whose entities are identified by a relationship with some other entity set, then the relation scheme also has the attributes in the key for the second entity set, but not its non-key attributes. Each tuple in the relation represents one entity in the entity set.

2. A relationship among entity sets  $E_1, E_2, \dots, E_k$  is represented by a relation whose relation scheme consists of the attributes in the keys for each of  $E_1, E_2, \dots, E_k$ . We assume, by renaming attributes if necessary, that no two entity sets have attributes with the same name. A tuple  $t$  in this relation denotes a list of entities  $e_1, e_2, \dots, e_k$ , where  $e_i$  is a member of set  $E_i$ , for each  $i$ . That is,  $e_i$  is the unique entity in  $E_i$  whose attribute values for the key attributes of  $E_i$  are found in the components of tuple  $t$  for these attributes. The presence of tuple  $t$  in the relation indicates that the entities  $e_1, e_2, \dots, e_k$  are related by the relationship in question.

*Example 3.3:* Let us explore a database that records baseball players, the teams they played for, their batting averages and positions played. Before showing how data of this nature can be represented as relations, let us consider the entity-relationship diagram that represents the "real world" as it pertains to this example. The entities are

1. PLAYERS, with attributes

NAME  
HOME /\* place of birth \*/  
BDATE /\* date of birth \*/

attribute NAME is a key.

2. POSITIONS, with attributes

POSNAME /\* e.g., pitcher \*/  
POSNUMBER /\* 1 for pitcher, 2 for catcher ... \*/

Either attribute can serve as a key, but we shall take POSNAME as the key.

3. TEAMS, with attributes

FRANCHISE /\* explained below \*/  
CITY  
YEAR

The FRANCHISE attribute is a unique identifier we give to a baseball franchise. As a franchise is property, it has existence as an entity even when it moves to another city or changes its name (e.g., the Cincinnati "Reds" have been called the "Redlegs" for two different periods of their history). In all example cases, we use the current name of the franchise as the value of FRANCHISE. A *team* (as opposed to a franchise) is the collection of players, coaches, etc., working for a franchise in a given year. The key for the entity set TEAMS is FRANCHISE and YEAR.

4. BA, the set of batting averages. This entity set has one attribute, PCT, whose values are three digit decimal numbers between 0 and 1. Clearly, PCT is the key.

We shall also discuss the following relationships.

1. The relationship SEASON, between PLAYERS, TEAMS, and BA. Player  $p$ , team  $t$  and batting average  $b$  are related by SEASON if  $p$  played on team  $t$ , and his batting average was  $b$ . Notice that SEASON is a ternary relationship. It is many-one from PLAYERS and TEAMS to BA, in the sense that given a player and a team, that player has a unique batting average. Recall that a "team" exists for one particular year, so the typical player is on many teams and has many batting averages, but just one batting average per year.
2. The relationship PLAYS between PLAYERS and POSITIONS. This is a many-many relationship indicating what positions were played by a player, over the course of his career.

The entity-relationship diagram is shown in Fig. 3.3. We draw an arrow from SEASON to BA to indicate that SEASON is many-one from PLAYERS-TEAMS to BA.

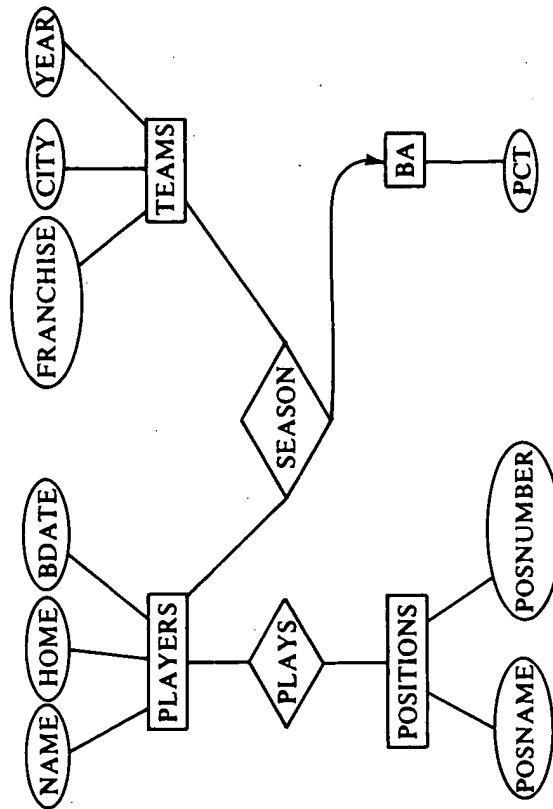


Fig. 3.3. The entity-relationship diagram for the baseball database.

Now let us select relation schemes to represent the entity sets and relationships. First, we have a relation scheme for each entity except BA. While there is no prohibition against a one-component relation (called a *unary relation*), such a relation would tell us nothing. It is merely a set

consisting of all the possible batting averages. The three relations for the other entity sets are:

- PLAYERS(NAME, HOME, BDATE)
- TEAMS(FRANCHISE, CITY, YEAR)
- POSITIONS(POSNAME, POSNUMBER)

The relationship PLAYS is represented by a relation whose attributes form the keys for entity sets PLAYERS and POSITIONS. Each of these sets has a one-attribute key, NAME and POSNAME, respectively. Thus we introduce a relation

PLAYS(NAME, POSNAME)

For relationship SEASON we need the keys of PLAYERS, TEAMS and BA. These are NAME, (FRANCHISE, YEAR), and PCT, respectively, so we also have the relation

SEASON(NAME, FRANCHISE, YEAR, PCT)

In Fig. 3.4 we see some sample tuples that would appear in these five relations if they contained all current data.<sup>†</sup> Notice that tuples need not appear in any particular order, and we certainly do not show all of them. For example, the SEASON relation has tuples for Ruth for all years between 1914 and 1935. □

Implementing a Relational Database

The obvious way to represent a relation is as a file whose record format consists of fields corresponding to the attributes in the relation scheme, in some particular order. Many data definition languages based on the relational model allow the user to specify, from among options, the organization of the file. The options available are usually a subset of those described in Sections 2.2 through 2.5, such as hashed or indexed. Another alternative, which may be the best for relations with few tuples, is the "heap," where the tuples are listed as records in the file in no particular order. A second useful feature found in many relational data manipulation languages is the ability for the user to specify secondary indices on certain attributes or sets of attributes.

As many file organizations are dependent on the existence of a key for records, a relational data definition language should also provide a mechanism for specifying one attribute or set of attributes that forms a key for the relation. The notion of a *key* for a relation is essentially the same as that of "key" in the context of files or entity sets. A relation is

<sup>†</sup> Source: Turkin and Thompson *The Official Encyclopedia of Baseball*, Barnes and Co.

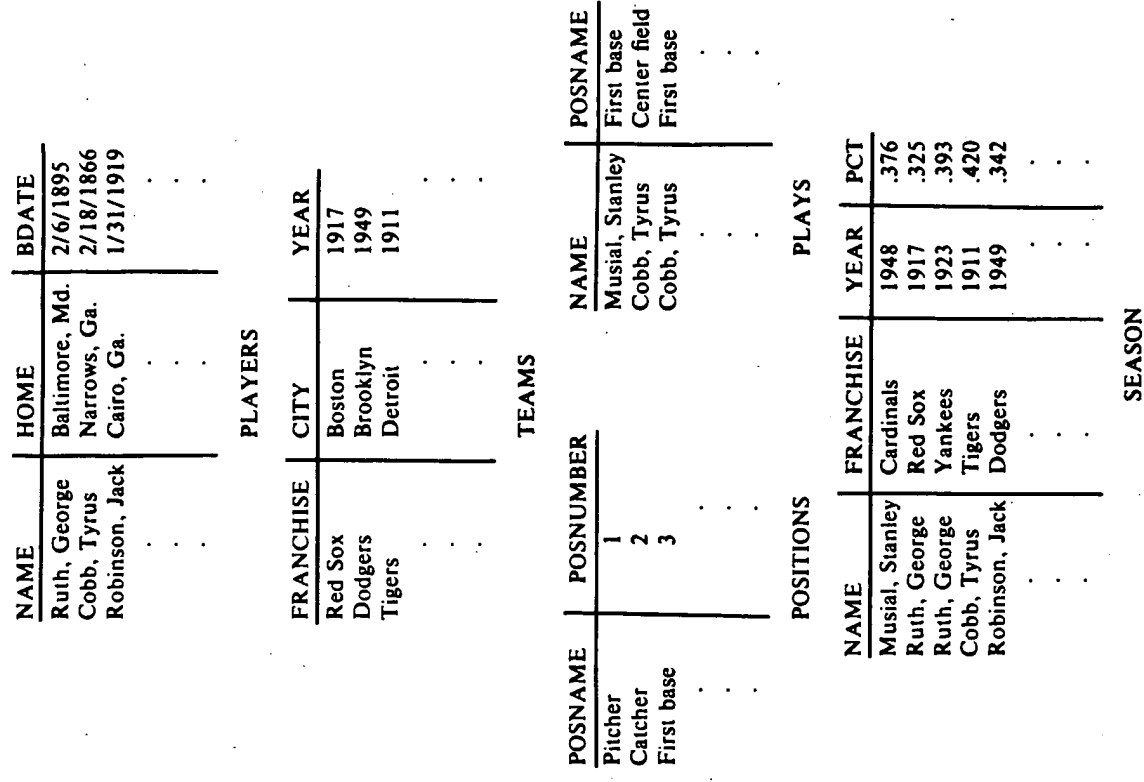


Fig. 3.4. Part of the five relations for the baseball database.



presumed not to have two tuples that agree on all the attributes of the key. For example, NAME and YEAR form a key for the relation SEASON in Fig. 3.4. We do not expect to find two tuples like

NAME	FRANCHISE	YEAR	PCT
Ruth, George	Red Sox	1917	.325
Ruth, George	Yankees	1917	.302

since we assume each player played for one team only in a given year (or in the case of a trade, we record only the team on which the player finished the season).

Clearly the attributes that form a key for the relation also serve as a key for the file. It is interesting to note that if we design our relation schemes according to the method outlined above, the relations for the entity sets adopt a key from the entity set. Also relations constructed to represent many-one relations from entity set  $E_1$  to set  $E_2$  can adopt the key of  $E_1$ . It is only relations for many-many relations that may have no key smaller than the trivial key consisting of all attributes. The general subject of finding and using keys in relations has been well developed, and we shall explore some of the theory in Chapter 5.

Let us emphasize that there is no requirement that each tuple be represented by a record, although this arrangement is by far the preferred implementation in existing systems. For example, a relation with attributes  $A$ ,  $B$ , and  $C$  could be represented by a logical file with variable length records of format  $A(B(C)^*)^*$ , and this logical file could be implemented in one of the ways suggested in Section 2.6. This arrangement saves space if there are relatively few values in the domain of attributes  $A$  and/or  $B$ . For example, the TEAMS relation of Example 3.3 could be represented conveniently as the logical file  $FRANCHISE(CITY(YEAR)^*)^*$ .  $FRANCHISE$  has the smallest domain, there being only two dozen or so franchises. In the logical file each franchise occurs exactly once, while in the relation TEAMS, each franchise occurs in a number of tuples equal to the number of years of its existence. Similarly, in the logical file, each city in which the franchise has resided appears once, while in relation TEAMS, each city appears in one tuple for each year the franchise was located in that city.

### Operations on Relational Databases

The basic operations by which a database is modified are the insert, delete, and modify instructions, essentially as introduced in Chapter 2. If the relational data model is used to describe the database, it is natural for these operations to apply to tuples. We shall describe the implementation of these three operations on the assumption that relations are represented by files in which tuples and records are in one-to-one correspondence, and the file is organized by one of the methods of Chapter 2. However, the reader

should be able to supply the details of implementation if logical files of variable length records are used. If yet another implementation is used, it is the implementor's responsibility to see that insert, delete and modify can be executed without undue difficulty.

1. *Insert.* For this operation, we are given the tuple to insert and the relation into which it is to be inserted. Select from the tuple the values of those attributes that form a key for the file. With these values, we have only to follow the directions given in Chapter 2 for the appropriate file organization, to insert the record made from the tuple. Conversion of a tuple to a record is not hard. As a tuple is a logical data item it may, for example, have for an attribute value a character string that is too short for the corresponding field of the record, which holds character strings of fixed length. Therefore we must pad the attribute value with blanks to make a field value. Similarly, the value of an attribute could be an integer while the corresponding field takes real values; again a straightforward conversion is necessary.
2. *Delete.* Suppose we are given a relation and values for at least those attributes that form a key for that relation and its underlying file. We may therefore apply the deletion procedure for the appropriate organization directly to the underlying file. Many systems allow deletion of a set of tuples even if the key is not given. In this case we must scan the entire file no matter which of the standard organizations are used.
3. *Modify.* Here we are given a relation, values for the key attributes and new values for those attributes that must change. Again the translation from attribute values to field values is straightforward, and we may apply the appropriate modification procedure to the file as described in Chapter 2. If we are not given values for attributes in the key, we are again forced to scan the entire file.

### Interrogation of a Relational Database

The lookup operation on a relation also presents no surprises. However, data manipulation languages using the relational model generally have a rich set of commands, from which we can build a variety of "queries" that obtain data from one or more relations. Some examples of how queries are expressed in existing relational database management systems will be given in the next chapter. However, let us here give some idea of typical queries and the problems they entail.

*Example 3.4:* Consider the relational database that was described in Fig. 3.4. One type of query asks for the values of certain attributes in those tuples of a relation that satisfy a particular condition. For example, we might ask for the name and franchise of all those players that batted .300 or over at least one season while playing for that franchise. To implement this query

we must scan the tuples of the SEASON relation, or strictly speaking, we scan the records of the file representing the relation. Each time we encounter a tuple for which PCT is at least .300, we collect the NAME and FRANCHISE values from that tuple. To avoid examining the record for every tuple, it would be useful to have a secondary index on attribute PCT for the file. We would also find it convenient if the secondary index, which is a logical file of variable length records of the form PCT(RECORD)\*, were sorted on its key, PCT. However, none of these conveniences are essential; they merely speed up processing.

Once we have collected all the desired pairs (NAME, FRANCHISE), we could print them out as is. Strictly speaking, we have produced a relation with relation scheme (NAME, FRANCHISE), and the mathematical notion of a relation does not permit duplicate tuples. Therefore, we should, in principle, remove duplicate tuples, which will occur whenever a player batted .300 or more for two or more seasons for the same franchise. We could, for example, sort the relation by NAME and, within each group of tuples with the same NAME, sort by FRANCHISE. Then duplicate tuples will be adjacent in the order, and can be removed easily. However, many relational DBMS's will not eliminate duplicate tuples unless specifically directed to do so by the user.

As a further example, let us again consider the database of Fig. 3.4, and suppose we ask for the franchise for which Ty Cobb played in 1911. Since NAME and YEAR form a key for the SEASON relation, it is straightforward to execute a lookup on the file representing SEASON, finding the desired record, and determining that the FRANCHISE attribute for the record is "Tigers."

Now suppose that instead of asking for Cobb's franchise in 1911, we asked for the city in which he played that year. This information is not available from the SEASON relation. However, we can obtain the answer by proceeding as above to determine that Cobb played for the Tigers in 1911, then using ("Tigers", 1911) as a key value in the TEAMS relation to discover that the Tigers franchise was in Detroit in 1911. □

The process of following logically connected data from relation to relation in order to obtain desired information is called *navigation*. We saw our first example of navigation in the last part of Example 3.4. In the next chapter we shall consider some of the ways that relational data manipulation languages allow the user to navigate, that is, to connect information from two or more relations. In general, navigation in a broader sense, meaning to follow connections between several parts of that database, is part of any data manipulation language, regardless of the model.

### 3.2 The Network Data Model

Roughly, the network data model is the entity-relationship model with all relationships restricted to be binary, many-one relationships.<sup>†</sup> This restriction allows us to use a simple directed graph model for data. It also makes a implementation of relationships simpler, as we shall see when we discuss a concrete implementation of the network model in Chapter 7.

There is no consistent terminology for the network model, so we shall adopt our own, trying to be consistent with terms we have used, or shall use, for files and for the other data models. In place of entity sets, we shall talk about *logical record types*, in the network model. A logical record type is essentially a relation, that is, a named set of tuples. However, as in the entity-relationship model, we admit the possibility that two identical records of the same logical record type exist; these records are distinguished only by their relationship to records of another logical type. In the network model, we use the term *logical record* in place of "tuple," and *logical record format* ‡ in place of "relation scheme." We call the component names in a logical record *format fields*.

Let us at the outset justify our decision to change terminology between the relational and the network model by reminding the reader that in the network model, logical record types are used principally to represent what we have called entity sets, while in the relational model, we use relations to represent both entities and relationships. While we cannot rule out bizarre uses of any data model, we feel it would do violence to the intuitive purposes of relations and logical record types if we tried to merge the two concepts.

Instead of "binary many-one relationships" we talk about *links*. We draw a directed graph, called a *network*, which is really a simplified entity-relationship diagram, to represent record types and their links. Nodes correspond to record types. If there is a link between two record types  $T_1$  and  $T_2$ , and the link is many-one from  $T_1$  to  $T_2$ , then we draw an arc from the node for  $T_1$  to that for  $T_2$ ,§ and we say the link is from  $T_1$  to  $T_2$ . Nodes and arcs are labeled by the names of their record types and links.

<sup>†</sup> Some rather general definitions of the network model allow many-many relationships, requiring only that they be binary.

<sup>‡</sup> We drop the word "logical" from "logical record," or "logical record type/format" whenever there is no confusion with the terms for files.

<sup>§</sup> Some works on the subject draw the arc in the opposite direction. However, we chose this direction to be consistent with the notion of functional dependency discussed in Chapter 5. Our point of view is that arrows mean "determines uniquely." Thus, as each record of type  $T_1$  is linked to at most one record of type  $T_2$ , we draw the arrow into  $T_2$ .

### Representing Entity-Relationship Diagrams in the Network Model

As we have stated, entity sets are represented directly by logical record types. The attributes of an entity set are fields of the logical record format. In the case that an entity is determined uniquely only through a relationship with another entity, we shall add another field that is a serial number for the entity set, uniquely identifying each entity. This serial number might be a field only on the logical level; in the implementation we could use the location of the record representing the entity as its "serial number."

Among relationships, only those that are binary and many-one (or one-one as a special case) are representable directly by links. However, we can use the following trick to represent arbitrary relationships. Say we have a relationship  $R$  among entity sets  $E_1, E_2, \dots, E_k$ . We create a new logical record type  $T$  representing  $k$ -tuples  $(e_1, e_2, \dots, e_k)$  of entities that stand in the relationship  $R$ . The format for this record type might consist of a single field that is a serial number identifying logical records of that type, although there are many situations where it is convenient to add other information-carrying fields in the format for the new record type  $T$ . We then create links  $L_1, L_2, \dots, L_k$ . Link  $L_i$  is from record type  $T$  to the record type  $T_i$  for entity set  $E_i$ . The intention is that the record of type  $T$  for  $(e_1, e_2, \dots, e_k)$  is linked to the record of type  $T_i$  for  $e_i$ , so each link is many-one.

**Example 3.5:** Let us represent in the network model the information about baseball players and teams shown in the entity-relationship diagram of Fig. 3.3. First we shall have logical record types PLAYERS, TEAMS and POSITIONS. The fields of their logical record formats are the same as the attributes of the relations of the same name, from Example 3.3. There is no logical record type for entity set BA, for reasons we shall discuss in connection with the relationship SEASON.

Consider now the many-many relationship PLAYS, between PLAYERS and POSITIONS. To represent PLAYS we need a new record type, which we call PP. The record format for PP consists of a serial number PPID. There are two links, PP\_PLAYERS from PP to PLAYERS, and PP\_POSITIONS, from PP to POSITIONS.

We also need to represent the ternary relationship SEASON among entity sets PLAYERS, TEAMS, and BA. We create a new logical record type PTB with a serial number field named PTBID. We create links PTB\_PLAYERS, from PTB to PLAYERS, and PTB\_TEAMS, from PTB to TEAMS. We could also create a link from PTB to BA, but we do not need to do so, because the SEASON relationship uniquely determines a batting average, given a player and team. Thus we could include the PCT attribute in the PTB record format, allowing us to avoid the existence of a BA record type and a link from PTB to BA. Notice that we could, in principle, include

the attributes of PLAYERS and TEAMS in the PTB format, as well, but to do so would waste a considerable amount of space; the attribute values for each team would be repeated once for each player on the team, and the attributes of a player would be repeated once for each team he was on.

The logical record types we have defined are listed below. As for relation schemes, we use the notation  $R(A_1, A_2, \dots, A_k)$  for record type  $R$  with format  $A_1, A_2, \dots, A_k$ .

PLAYERS(NAME, HOME, BDATE)  
TEAMS(FRANCHISE, CITY, YEAR)  
POSITIONS(POSNAME, POSNUMBER)  
PP(PPID)  
PTB(PTBID, PCT)

The links are:

PP\_PLAYERS from PP to PLAYERS  
PP\_POSITIONS from PP to POSITIONS  
PTB\_PLAYERS from PTB to PLAYERS  
PTB\_TEAMS from PTB to TEAMS

The network is shown in Fig. 3.5.

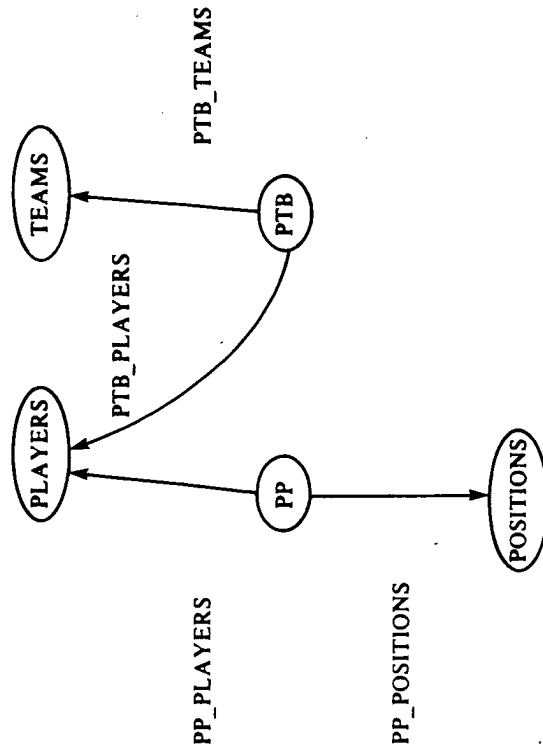


Fig. 3.5 Network for baseball database.

In Fig. 3.6 we show some sample logical records of each type and the occurrences of links among these records. There may, of course, be other link occurrences involving some of the records shown. □

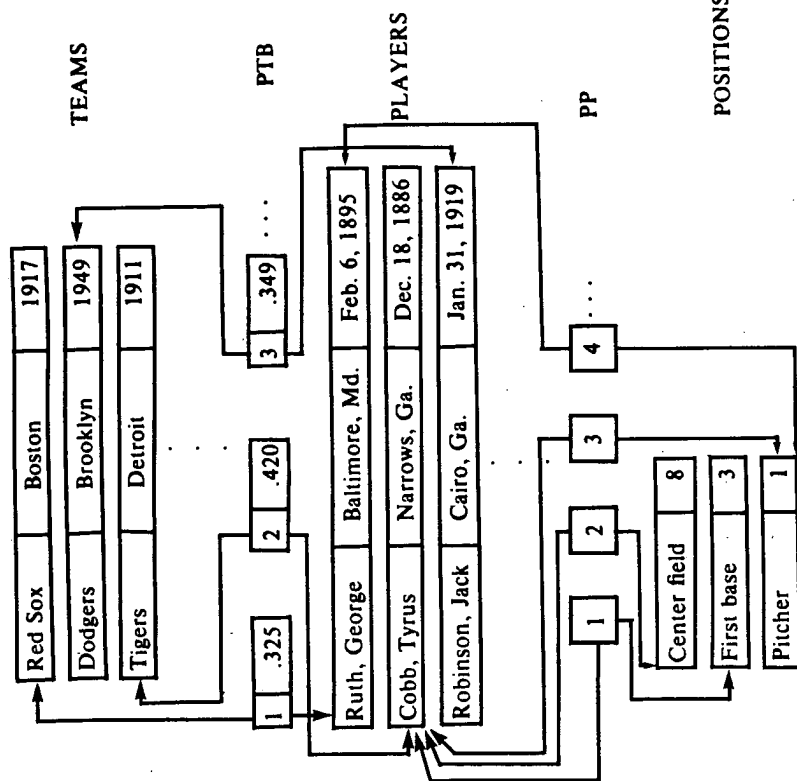


Fig. 3.6. Some logical records in the baseball database.

### Implementation of a Network

We can represent the logical records of a given record type by a file in the obvious manner, with one field of each record for each field of the logical record. We may dispense with fields for artificially created serial number attributes as in PP or PTB of Example 3.5, by using the location of the records to identify records uniquely. As we shall see, even if a logical record format consists only of the serial number attribute, its records need not disappear entirely, and the presence of those records surely will influence the organization of the database, no matter what representation we choose for the network.

Suppose we have a link from record type  $T_1$  to record type  $T_2$ . Since the link is many-one from  $T_1$  to  $T_2$ , we could represent it by variable

length records of the format  $T_2(T_1)^*$ . That is, after each record of type  $T_2$ , list all the associated records of type  $T_1$ . The variable length records can then be represented in one of the ways discussed in Section 2.6. If there is another link from record type  $T_3$  to  $T_2$ , we can list the occurrences of  $T_3$  records with the corresponding  $T_2$  records, using a variable length record format like  $T_2(T_1)^*(T_3)^*$ . Again, the methodology of Section 2.6 can be used to implement such variable length records.

However, suppose there is another link from  $T_1$  to some record type  $T_4$ . We cannot list  $T_1$  records after  $T_2$  records and also list them after  $T_4$  records, or at least, it would hardly be efficient or convenient to do so. We therefore need another way of representing links, one that does not force records of one type to be adjacent to records of another type. In this organization, called a *multilist*, each record has one pointer for each link in which it is involved, although there is the option of eliminating the pointer for one link and representing that link by variable length records as discussed above.

Suppose we have a link  $L$  from  $T_1$  to  $T_2$ . For each record  $R$  of type  $T_2$  we create a circular chain from  $R$ , to all the records  $R_1, R_2, \dots, R_k$  of type  $T_1$  linked to  $R$  by  $L$ , and then back to  $R$ . The pointers for link  $L$  in records of types  $T_1$  and  $T_2$  are used for this purpose. We show an example chain in Fig. 3.7. Note how we can follow the chain from  $R$  to visit each of  $R_1, R_2, \dots, R_k$  and, if records of type  $T_2$  are identifiable in some way (e.g., each record begins with a few bits indicating its record type, or the record address determines the type) we can go from any of  $R_1, R_2, \dots, R_k$  to  $R$ .

It is important to remember that in a multilist organization, each record has as many pointers as its record type has links. As the pointers are fields in the records, and therefore appear in fixed positions, we can follow the chain for a particular link without fear of accidentally following some other link. Also remember that since links are many-one, each circular chain has exactly one record of one type and zero or more records of the second type. Notice that if we tried to represent many-many relationships by multilists we would face severe problems, since each record could be on many chains for the same relationship. We would not know in advance how many pointers were needed in a record for each link, and we would have trouble determining which link was for which chain. The problem is not conveniently solved, which explains why the network model emphasizes many-one links.

**Example 3.6:** Let us take a simple example of a many-many relationship implemented by two many-one links and a dummy record type. The two entity sets involved in the relationship are DEPTS (of a department store) and ITEMS, and the relationship is SELLS. As most departments sell more than one item, and an item could be sold by more than one department,

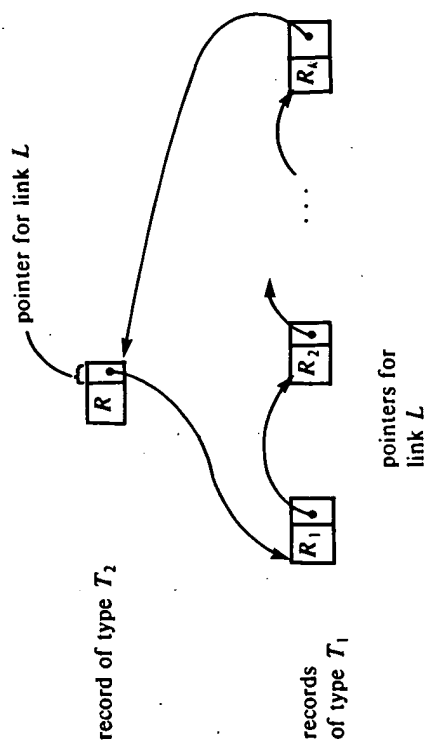


Fig. 3.7. A circular chain.

SELLS is a many-many relationship. To represent SELLS in the network model, we need the intermediate record type DI and two links:

1. SELLING\_DEPT from DI to DEPTS, and
2. ITEM\_SOLD from DI to ITEMS.

In Fig. 3.8 we see a portion of the physical database, where links are represented by multilists. We see two departments, Clothing and Toys, and five items,

1. Shirts, sold by the Clothing Dept.
2. Skirts, sold by the Clothing Dept.
3. "G.I. Jack" dolls, sold by the Toy Dept.
4. G.I. Jack Flak Jackets, sold by both the Toy and Clothing Depts.
5. Basilisk collars, sold by no department, due to insufficient demand.

We see in Fig. 3.8 two pointers in each DI record; the first is for link SELLING\_DEPT and the second for ITEM\_SOLD. The DEPTS records show only the pointer for SELLING\_DEPT; there could be other links involving DEPTS. Similarly, we show in ITEMS records only the pointer for the ITEM\_SOLD link. SELLING\_DEPT pointers are shown with solid lines and ITEM\_SOLD pointers are dashed lines. □

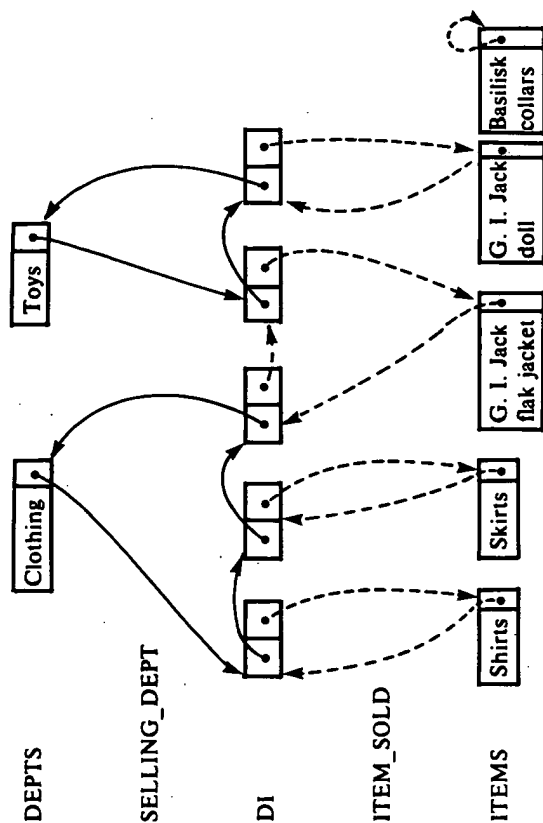


Fig. 3.8 A multilist implementation of a network.

### Operations on Networks

As in the relational model, the basic commands to change the database are insertions, deletions and modifications. But while in the relational model we had only to consider these operations on relations, for the network model we must consider the operations on both logical record types and on links. The principles are the same for all three operations, in that each operation is implemented by the corresponding operation on a file, so let us discuss only insertion.

To insert a logical record into a logical record type, we create a physical record from the logical record just as we converted a tuple to a record in the relational model. We then insert the record into the file representing the logical record type, using the appropriate technique from Chapter 2, depending on the file organization.

Suppose now that we have a link  $L$  from record type  $T_1$  to record type  $T_2$ , and we wish to add to this link the fact that record  $R_1$  of type  $T_1$  is associated with record  $R_2$  of type  $T_2$ . First, if necessary, we insert  $R_1$  and  $R_2$  into the files for  $T_1$  and  $T_2$ , respectively. If link  $L$  is represented by variable length records of the format  $T_2(T_1)^*$ , we look up the record for  $R_2$  in the file representing the variable length records, and insert  $R_1$  onto the list of associated records of type  $T_1$ , using an algorithm from Section 2.6; the exact algorithm depends on how variable length records are represented.

If link  $L$  is represented by a multilist organization, we again look up the record for  $R_2$  in the file for  $T_2$ . We also look up the record for  $R_1$  in the file for  $T_1$ . We then link the latter onto the chain of the former, using elementary list processing operations on the fields of  $R_1$  and  $R_2$  that hold the pointers associated with  $L$ .

Note that for both the above representations of links we assume that  $R_1$  was not previously linked to any record of type  $T_2$ . If that assumption is false we must first look up  $R_1$  in the file for  $T_1$ . If the multilist strategy is used it is a simple matter to remove  $R_1$  from whatever chain for  $L$  on which  $R_1$  appears. If the variable length record approach is used to represent  $L$ , matters are not so simple. There may be no convenient way to get from  $R_1$  to the record  $R_3$  of type  $T_2$  with which it is currently linked. Fortunately, an easy fixup for each of the representations of variable length records exists to allow us to get from  $R_1$  to  $R_3$ . For example, if the records of type  $T_2$  have a pointer to a chain of blocks with associated records of type  $T_1$ , we can end the chain with a pointer back to the type  $T_2$  record to which the chain belongs.

### Queries on Networks

The simplest queries to answer about a database represented by the network model, involve only attributes of one record type. For example, referring to the database of Example 3.5, we could answer: "What is Ruth's birthday?" by doing a lookup on the file for record type PLAYERS, with value "Ruth, George" for the attribute NAME. Since NAME is the key for PLAYERS, this lookup can be done efficiently. If the query gives us the values of attributes that do not form a key, then we need secondary indices for the file on at least some of the given attributes, or we must search the entire file.

Next, in order of difficulty, come queries that involve following a link. We may be given a key value for a record of type  $T_2$  and be asked to find a record or records of type  $T_1$  to which it is linked by link  $L$  from  $T_1$  to  $T_2$ . Referring again to Example 3.5, we may be asked "Did Robinson ever bat over .340?" We find the PLAYERS record for Robinson, which gives us access to the list of PTB records for Robinson, by following the PTB\_PLAYERS link. We can scan these records for one with a PCT value above .340 and find one with .342.

It is also possible that we are given a key value for a record  $R$  of type  $T_1$  and are asked to find the unique record of type  $T_2$  linked to it by link  $L$  from  $T_1$  to  $T_2$ . If  $L$  is represented in multilist fashion, this task is easy; we follow the chain for  $L$  on which  $R$  appears, until we come to the record of type  $T_2$  on the chain. If  $L$  is represented by variable length records, we must have available a pointer that gets us from  $R$  to the beginning of the variable length record of which it is part, as discussed in connection with

the deletion of links. If the pointer exists, we have no problem; if not we shall have to scan the entire file of records of type  $T_2$  to discover which one has  $R$  associated with it.

The hardest kind of query to handle is one that involves following two or more links. For example, suppose we ask: "What center fielders batted over .400?" We begin by finding the POSITIONS record for "Center field," which is easy, since POSNAME is a key for that record type. Next we examine each of the PP records linked to "Center field" by the PP\_POSITIONS link. We follow the PP\_PLAYERS link to obtain a unique PLAYERS record for each PP record. Lastly, from each PLAYERS record found, we examine the PTB records linked to it by the PTB\_PLAYERS link, and if one or more PTB records for that player have a PCT value above .400, we print the name of the player.

### 3.3 The Hierarchical Data Model

A *hierarchy* is simply a network that is a *forest* (collection of trees) in which all links point in the direction from child to parent. We shall continue to use network terminology "logical record type," and so on, when we speak of hierarchies. One additional concept that is useful when dealing with hierarchies is the *virtual* logical record type, which is a pointer to a record of a given logical type. Virtual record types are needed in situations where intuitively we would like to place a record type in two or more trees of a hierarchy, or even in several places in one tree. We do not wish to place two copies of a record in two positions in the database; to do so wastes space and invites the possibility that we might change one copy without changing the other. Therefore, each logical record type appears in one place in the hierarchy, and other places where we would like that record type are given virtual records instead.

Using virtual types, we can convert any network to a hierarchy. Begin with any logical record type  $R$ , which becomes the root of the first tree. If possible, choose  $R$  to have no links leaving. The children of record type  $R$  are any types that have links entering  $R$ . Their children are found by following links backward from them, from head to tail, and so on. However, if we ever encounter a type we have already placed in the hierarchy, we create a virtual record type, put this virtual record type in the hierarchy instead of the logical type encountered, and do not add any children of the virtual type. When we can add no more children to the tree under construction, look for a logical record type not already placed in the hierarchy. If we find none, we are done. Otherwise, repeat the tree construction process with one of the previously unplaced logical record types.

*Example 3.7:* Let us convert the network of Fig. 3.5 to a hierarchy. We might begin with PLAYERS, which has arcs from PP and PTB entering, but no arcs leaving. The latter two record types have no arcs entering, so

we are done with the first tree; it has PLAYERS as root and PP and PTB as children. Our second tree might start with TEAMS. There is one entering arc, from PTB, which we have already placed in the hierarchy. Thus we give TEAMS a virtual record type "pointer to PTB" as child and go on to the third tree, which consists of root POSITIONS with virtual PP as child. The entire hierarchy is shown in Fig. 3.9. Let us observe now that this hierarchy is a poor way to represent the baseball database, because the sorts of queries about this database that are phrased easily in hierarchy-based data manipulation languages do not include some very natural queries. We shall justify this remark about queries and give a better hierarchical representation shortly.

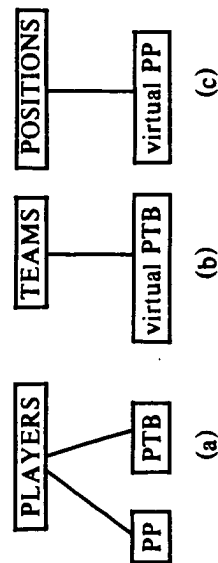


Fig. 3.9. A hierarchical representation of the baseball database.

An instance of the database described by a tree of logical record types may be viewed as a forest. For each logical record type in the hierarchy, the database will have zero or more nodes, each representing a record of that type. It is sometimes useful to draw each tree in the database with a dummy root, whose children are the record occurrences of the root record type. For example, part of the database tree for scheme (a) of Fig. 3.9 is shown in Fig. 3.10. The numbers 1 through 5 are serial numbers, which identify PP and PTB records. The serial numbers disappear in an implementation. □

#### Direct Representation of Many-Many Relationships in a Hierarchy

Sometimes, hierarchies designed directly from a network can lead to awkwardness when we try to answer queries. Consider the hierarchy of Fig. 3.9, and suppose we wanted to know what positions Cobb played. We would start at the root of Fig. 3.10 and search for the PLAYER record for Cobb. We might be aided in our search by an appropriate organization of the file of PLAYER records, so suppose we find the Cobb record easily. We scan those children of that record that are of PP type. Say we examine the record with serial number 1. We must now go to the database corresponding to tree (c) of Fig. 3.9 and examine each POSITION record. For each such record, we look at its children, which are virtual PP records, that is, pointers to PP records in the database for tree (a). If we find a

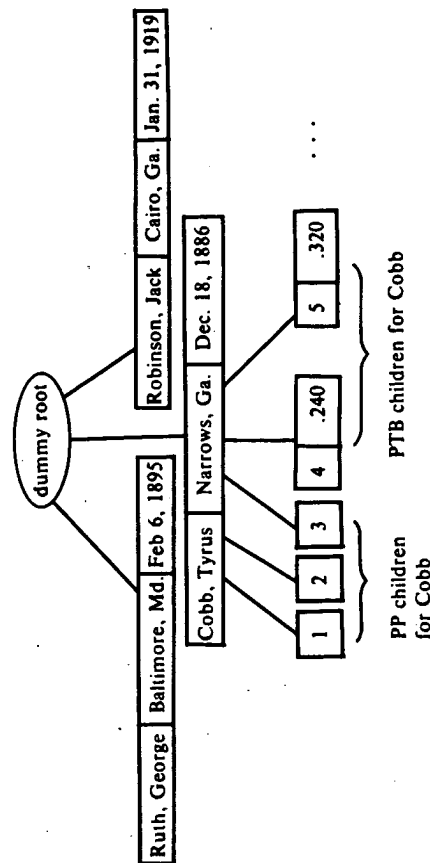


Fig. 3.10. Part of a database in the form of a tree.

virtual PP record for a position that points to the PP record with serial number 1, then we know Cobb played that position.

We might ask why it is much easier to answer this query in the network of Fig. 3.5 than in the hierarchy of Fig. 3.9. The reason is that links in a network are essentially two-way connectors, and in Fig. 3.5 we are able to go directly from a PP record to a POSITIONS record. However, the algorithm above for converting a network to a hierarchy broke the link from PP to POSITIONS, although the link from POSITIONS to PP exists due to the pointers in the virtual PP records. There is an inherent tendency for links to get broken when we go from a network to a hierarchy, although they can be maintained by introducing extra pointer fields into the logical record types of the hierarchy.

Another problem with a poorly designed hierarchy is that while we can navigate wholly within a database built along the lines of the network data model, in a hierarchy we must sometimes jump from tree to tree. This can only be accomplished by reading a value from the database into the workspace of the application program (e.g., the serial number 1 in Example 3.7) and using that value to access information in another tree.

For these reasons, one must often exercise care in designing a hierarchy, so values will be available where needed. One useful idea is the following. Suppose we have entity sets  $E_1$  and  $E_2$  with a many-many relationship  $R$  between them. We can represent  $R$  in a hierarchy by selecting either  $E_1$  or  $E_2$ , say  $E_1$ , to be the root of a tree. We give the root a child logical record type  $T$  consisting of

1. a serial number, to identify records, and
2. attributes forming a key for  $E_2$ .

As in networks, the serial number might not appear in the implementation, being replaced, in essence, by the address of the record. We do not wish to place all attributes of  $E_2$  in the format for record type  $T$ , because we would then have the task of making sure that several  $E_2$  records representing the same entity had the same information. Instead, we create another logical record type with all the attributes of  $E_2$ ; this record type will be placed somewhere in the hierarchy — at the root of its own tree if no more convenient place can be found.

In some applications it is only necessary to find  $E_2$  entities associated with a given  $E_1$  entity, and if so, the above arrangement is ideal. However, in other cases it may also be essential that we be able to find the  $E_1$ 's associated with an  $E_2$  conveniently. If this is the case, we can include, as a child of the  $E_2$  record type, a new record type consisting of the key fields of  $E_1$ .

Another possibility is to replace the record types consisting of the keys for  $E_1$  and/or  $E_2$  by pointers to  $E_1$  or  $E_2$  records, respectively. This arrangement pins the records pointed to, but it allows direct access to those records. In comparison, using key values to indicate records requires us to search for the desired record. The reader is referred to Chapter 8 for a detailed example of how a major hierarchical DBMS facilitates the implementation and use of many-many relationships in practice.

*Example 3.8:* Let us redesign the hierarchy for the baseball database. Remember, the original definition of the database scheme is the entity-relationship diagram of Fig. 3.3. To express the PLAYS relationship we shall have a tree whose root is the logical record type

(NAME, HOME, BDATE)

representing PLAYERS. The root has a child logical record type

(SERIAL1, POSNAME)

representing the names of the positions each player plays. The attribute SERIAL1 is a serial number used to identify records, and it will probably disappear in an implementation, being represented by the location of the record.

To represent the association between position names and numbers there will be a second tree with only a root logical record type:

(POSNAME, POSNUMBER)

We could use the attributes of entity set TEAMS as the root of a third tree, but since we are designing a hierarchy, we may as well take advantage

of the fact that there is a hierarchical structure to the set of teams. We choose as root for the third tree a logical record type consisting only of FRANCHISE. A franchise can be viewed as composed of teams, one for each year, so the root of the third tree has one child, which is the logical record type

(SERIAL2, CITY, YEAR)

In the database itself, each record of the above type represents a team, which is determined by the value of YEAR and the value of FRANCHISE in the parent of the record. Again, SERIAL2 is a dummy serial number used to distinguish records of this type, since there may be two franchises in a given city in a given year.

The above logical record type has a child logical record type representing the players on a team and the batting average of that player for that year; its logical record format is:

(SERIAL3, NAME, PCT)

We expect SERIAL3, like the other serial numbers to disappear in an implementation, and it is present in the conceptual database scheme to prevent us from identifying records with the same NAME and PCT if they represent the batting average of a player for two different seasons. The entire hierarchy is shown in Fig. 3.11.

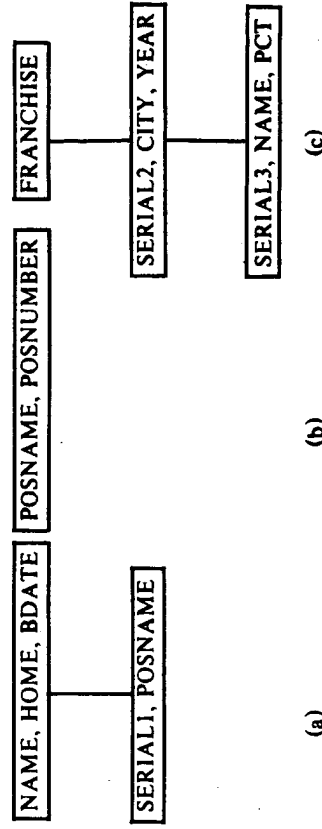


Fig. 3.11. An improved hierarchy for the baseball database.

In the hierarchy of Fig. 3.11, we can answer a query such as: "What positions did Cobb play?" by finding the Cobb record in the file for the root record type in a tree (a) and examining its children, each of which has the name of a position Cobb played. We could even find the numbers of the positions Cobb played by proceeding as above, then looking in tree (b) for the record for each position, to get the position's number. □



### Implementation of Hierarchical Databases

The ideas used in the implementation of a network database carry over to the hierarchical model directly, since hierarchies are special cases of networks. Representation by files of variable length records is especially suitable for hierarchies. To construct a variable length record format for a tree, use the following two rules, working up the tree.

1. The format for a leaf is  $(\alpha)^*$ , where  $\alpha$  is the list of attributes in the logical record format for this leaf.
2. If a node has  $k$  children with variable length record formats  $\alpha_1, \alpha_2, \dots, \alpha_k$ , and the list of attributes in the logical record format for this node is  $\beta$ , then the variable length record format for the node is  $(\beta\alpha_1\alpha_2 \dots \alpha_k)^*$ .

In following the above rules, we construct one file of variable length records for each tree in the hierarchy. Of course, each such file must be implemented, as discussed in Section 2.6, such as by one or more files of fixed length records.

*Example 3.9:* The variable length record format for the tree of Fig. 3.11(c) is

(FRANCHISE(SERIAL2 CITY YEAR (SERIAL3 NAME PCT))\*)\*

and for Fig. 3.9(a) it is:

(NAME HOME BDATE(SERIAL1)\* (SERIAL2 PCT))\*)\*

□

Another possible implementation of a hierarchy is to list the records of each tree in preorder as we suggested in Section 2.6 for fields of a variable length record format. A preorder traversal, or listing, of a tree is defined recursively, as follows.

1. list the root.
2. for each child  $c$  of the root, from the left **do**  
list the subtree with root  $c$ , in preorder

*Example 3.10:* Suppose we have the tree of Fig. 3.12. To list it in preorder, we list  $A$ , the root, and then list the tree of  $B$ ,  $D$ , and  $E$ , in preorder. To do so, we list  $B$  and then the tree consisting of  $D$  alone, then the tree of  $E$  alone. Thus the first four nodes in preorder are  $ABDE$ . To complete the preorder listing we list the tree whose root is  $C$ , which gives  $ABDECF$  as the complete preorder. □

When we list in preorder a tree of records in a database, we must identify each record by some bits at the beginning that indicate its logical record type. This is necessary so we know how to find and interpret the values of fields in the record. It is also convenient to make available in

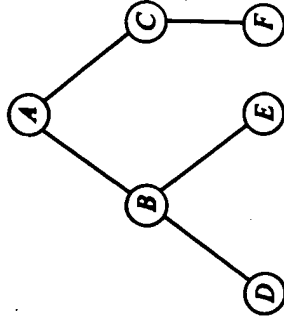


Fig. 3.12. A tree.

each block over which the records are spread, information telling where in the block each begins, since we do not know *a priori* what type each record is or how long it is. As we shall discuss in Section 8.4, it is also useful to thread the list with pointers of various sorts to provide easy access to siblings.

*Example 3.11:* We may view Fig. 3.12 as a conceptual schema described using the hierarchical model, where  $A, B, \dots$  represent logical record types. In Fig. 3.13 is a possible database represented by the schema of Fig. 3.12. We could list each of the records represented by a node in Fig. 3.13 in preorder, over as many blocks as needed, as follows:

$a_1b_1d_1d_2d_3e_1b_2d_4c_1f_1f_2c_2f_3a_2b_3d_5e_2c_3f_4f_5f_6c_4f_7$

Notice that the above string is an instance of the variable length record format  $(A(B(D)^*(E)^*(C(F)^*))^*(C(F)^*))^*$  constructed from Fig. 3.12. □

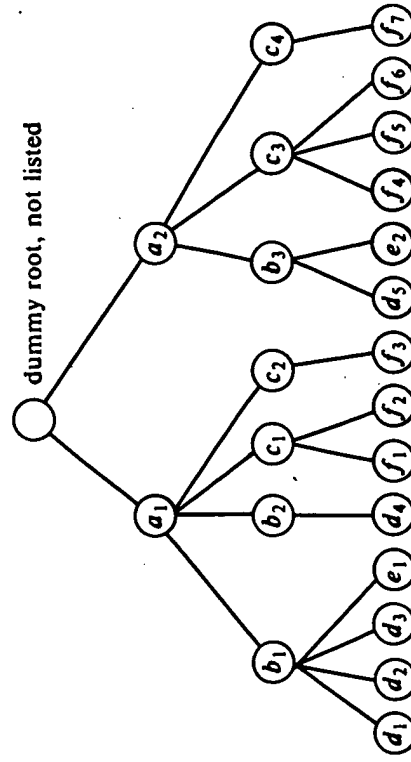


Fig. 3.13. A tree-structured database.

3.4 Comparison of the Models

To evaluate the three models discussed in this chapter we must first state the criteria by which they should be judged. We see two primary concerns.

1. *Ease of use.* Especially in small databases, on the order of thousands or tens of thousands of records, the principal cost may be the time spent by the programmer writing applications programs and by the user posing queries. We want a model that makes accurate programming and the phrasing of queries easy.
2. *Efficiency of implementation.* When databases are large, the cost of storage space and computer time dominate the total cost of implementing a database. We need a data model in which it is easy for the DBMS to translate a specification of the conceptual scheme and the conceptual-to-physical mapping into an implementation that is space efficient and in which queries can be answered efficiently.

By the criterion of easy use, there is no doubt that the relational model is superior. It provides only one construct that the programmer or user must understand. Moreover, as we shall see, there are rich, high-level languages for expressing queries on data represented by the relational model. These languages make systems based on the relational model available to persons whose programming skill is not great.

In comparison, the network model requires our understanding of both record types and links, and their interrelationships. The implementation of many-many relationships and relationships on three or more entity sets is not straightforward, although with practice one gets used to the technique, discussed in Section 3.2, of introducing dummy record types. Similarly, the hierarchical model requires understanding the use of pointers (virtual record types) and has the same problems as the network model regarding the representation of relationships that are more complex than many-one relationships between two entity sets.

When we consider the potential for efficient implementation, the network and hierarchical models score high marks. We saw in Section 3.2 how implementations of variable length records can facilitate the task of following links. We also mentioned that data structures such as the multilist and the pointer-based implementation of variable length records do not generalize readily to many-many mappings. Since relations can, and often do, represent many-many mappings, we see that efficient implementation can be more difficult for relations than for networks or hierarchies. Fortunately, some specialized data structures can be used to implement relations, as well. In Section 4.5 we shall discuss how the multilist and other ideas have been adapted to relations in one existing system.

Efficient utilization of space sometimes can be easier using hierarchies (or networks in the form of a hierarchy) than relations. As a case in point,

consider the hierarchy of Fig. 3.14(a) and an example database in Fig. 3.14(b). If we wished to represent the same structure by relations, we would probably choose one relation with attributes *A* and *B*, another with attributes *A* and *C*, and a third with attributes *C* and *D*. The corresponding relations are shown in Fig. 3.15. Observe how *a*<sub>1</sub>, *c*<sub>1</sub>, and *c*<sub>2</sub> are each repeated several times, resulting in a waste of space, compared with Fig. 3.14(b).

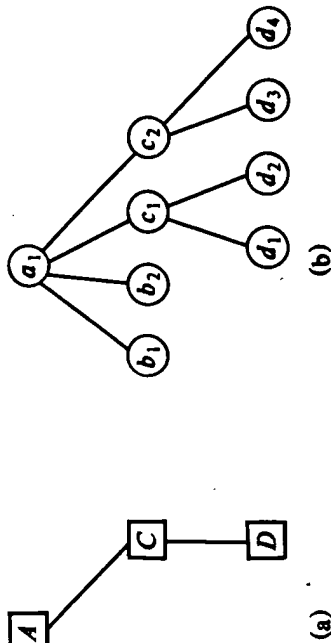


Fig. 3.14. A hierarchy and database.

A		B		A		C		D	
a <sub>1</sub>	a <sub>1</sub>	b <sub>1</sub>	b <sub>2</sub>	a <sub>1</sub>	a <sub>1</sub>	c <sub>1</sub>	c <sub>1</sub>	d <sub>1</sub>	d <sub>2</sub>
a <sub>1</sub>	a <sub>1</sub>	b <sub>2</sub>	b <sub>2</sub>	a <sub>1</sub>	a <sub>1</sub>	c <sub>1</sub>	c <sub>2</sub>	d <sub>2</sub>	d <sub>3</sub>
						c <sub>2</sub>	c <sub>2</sub>	d <sub>3</sub>	d <sub>4</sub>

Fig. 3.15. Relations.

The above effect would not be important if it were not that data frequently forms a natural hierarchy. While we are free to implement relations by a hierarchical structure such as Fig. 3.14(b), it is not always clear how to do so. Consider, for example, what the hierarchy corresponding to Fig. 3.15 would look like if *C* were chosen as the root of the conceptual scheme, rather than *A*.

Language Level

The level of the data manipulation language can affect profoundly the ease with which a DBMS can be used, just as it is easier to program in FORTRAN than in assembly language, and easier to program in APL than in FORTRAN. It has been the case that relational DBMS's have stressed languages of very high-level, while DBMS's based on the other models have tended to have languages of lower-level. We shall see in Chapter 4

some of the high-level relational languages, while Chapters 7 and 8 exhibit two of the principal low-level languages in which navigation details are expressed in the network and hierarchical models.

One might wonder if there is an inherent reason why these examples are typical. It is hard to argue that there is, although we should point out that the principal general-purpose languages of very high level, such as APL, SNOBOL, LISP, or SETL, are each based on a single data type (the array, string, list structure, and set, respectively), just as the relational model provides but a single data type, the relation. In contrast, lower level languages, like PL/I, have a variety of data types at the programmer's disposal.

### Conclusions

In the past, commercial database systems have almost uniformly been based on the network or hierarchical model, because the emphasis of such systems has been on the maintenance of large databases, and these models lend themselves most easily to the necessary efficient implementation. However, it is felt that the relational model will receive the bulk of attention in the future for two reasons. First, it is becoming clearer that the same concepts used to design large databases apply as well to small and medium scale databases, and there are many more small (i.e., thousands of records rather than millions) databases than large ones. With small databases, the ease of use inherent in the relational model assumes increased importance.

Second, many of the apparent inefficiencies of the relational model can be eliminated. Chapter 6 discusses some of the optimization techniques for relational data manipulation languages that allow these languages to use time efficiently. Research aimed at producing good physical implementations of relations is also underway. We shall discuss some of the progress connected with IBM's System R in Section 4.5, and Section 4.7 discusses Query-by-Example, an existing commercial system suitable for medium size databases. Just as general-purpose very high level languages such as APL are winning converts as the rather difficult optimization problems associated with these languages gradually are solved, we suspect that the relational model will become more attractive as the optimization problems associated with this model are likewise solved.

### Exercises

- 3.1: In Fig. 3.16 we see the entity-relationship diagram of an insurance company. The keys for EMPLOYEES and POLICIES are EMP# and P#, respectively; SALESMEN are identified by their *isa* relationship to EMPLOYEES. Represent this diagram in the (a) relational (b) network (c) hierarchical models.

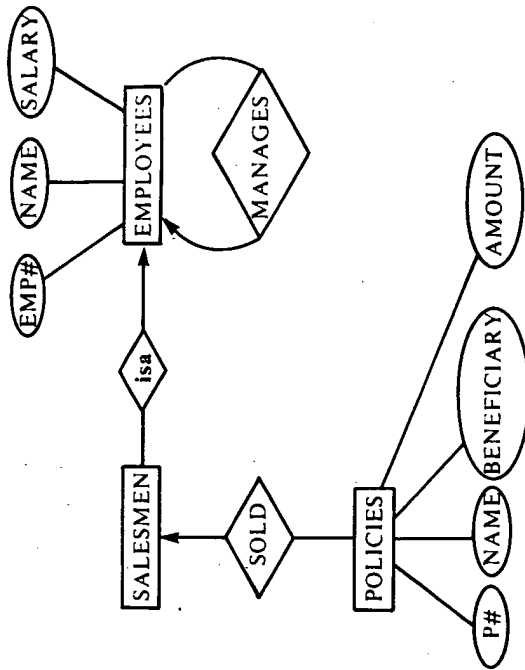


Fig. 3.16. An insurance company database.

3.2:

Figure 3.17 shows a genealogy database, with key attributes NAME and LIC#. Represent this diagram in the (a) relational (b) network, and (c) hierarchical models.

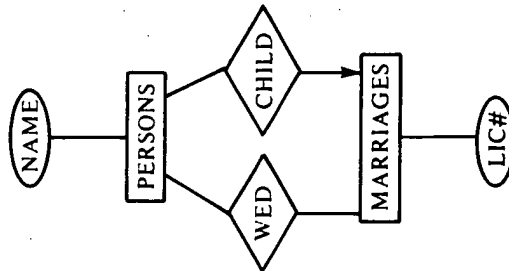


Fig. 3.17. A genealogy database.

3.3: The recipe for *moo shoo roe* includes bamboo shoots, sliced pork, wood ears, golden needles, and assorted vegetables. *Hot and sour soup* is made from wood ears, bean curd, and golden needles, while *family style bean curd* is made from bean curd, sliced pork, and assorted vegetables.

- Suppose we wish to store this information in a relation RECIPE(DISH, INGREDIENT). Show the current value of the relation as a table (use suitable abbreviations for the dishes and ingredients).
- Suppose we wish to represent the above information as a network with record types DISH, INGREDIENT and DUMMY, where a DUMMY record represents a pair consisting of one ingredient for one dish. Suppose also that there are links USES from DUMMY to DISH and PART\_OF from DUMMY to DISH and INGREDIENT. Draw the DISH, INGREDIENT, and DUMMY record occurrences and represent the links USES and PART\_OF by the multilist structure.
- Do the same as in part (b), but represent USES by variable length records with format DISH(DUMMY)\*; represent PART\_OF by a multilist structure.
- Suppose we use the hierarchy of Fig. 3.18 as a conceptual scheme for the above information. Show the actual database, including pointers in the virtual records.

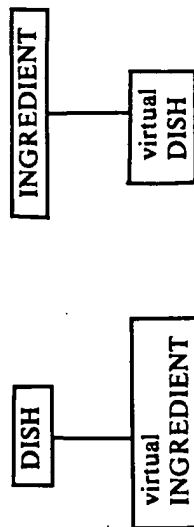


Fig. 3.18. A hierarchy.

\*3.4: While database systems generally have their own specialized data manipulation languages, we can get a feel for how our three models facilitate querying the database if we use a conventional language like PL/I† to set up a database and write queries.

- Write PL/I record structure declarations for the relations from Exercise 3.3(a).

† Any language with record structures, such as PASCAL or C will do as well.

- Write a PL/I program to print all the dishes using wood ears, using the structure from (a).
  - Write PL/I record structures to represent the record types from Exercise 3.3(b). Include fields for the pointers needed to represent the multilist structure.
  - Write a PL/I program to print all the dishes using wood ears. Start by finding the ingredient record for wood ears, and search its ring for the PART\_OF link.
  - Write record structures for the hierarchy of Fig. 3.18. You must find a way of representing the children of DISH and INGREDIENT records, for example, by creating linked lists of virtual INGREDIENT and virtual DISH records.
  - Write a program to print all the dishes using wood ears, using the structure from (e).
- \*3.5: We mentioned in Section 3.1 that two tables represent the same relation if one can be converted to the other by permuting rows and/or columns, provided the attribute heading a column moves along with the column. If a relation has a scheme with  $m$  attributes and the relation has  $n$  tuples, how many tables represent this relation?

#### Bibliographic Notes

The reader should see the bibliographies of Chapters 4, 7, and 8 for references to a variety of systems that use the relational, network, and hierarchical models, respectively. We previously mentioned in Chapter 1 a variety of other models for conceptual schemes; these are generally at a somewhat higher level than the three of this chapter and are intended primarily as database design tools to be translated into one of the three covered here.

Of the three models, only the relational has its origin in theoretical proposals predating any true implementations. The concept is attributed to a series of papers by Codd [1970, 1972a, 1972b] although Codd [1970] acknowledges some earlier work of Childs [1968]. Several works have been devoted to comparisons of the three models, of which we note the compendium Rustin [1974] and the issue of *Computer Surveys*, Sibley [1976].

some of the high-level relational languages, while Chapters 7 and 8 exhibit two of the principal low-level languages in which navigation details are expressed in the network and hierarchical models.

One might wonder if there is an inherent reason why these examples are typical. It is hard to argue that there is, although we should point out that the principal general-purpose languages of very high level, such as APL, SNOBOL, LISP, or SETL, are each based on a single data type (the array, string, list structure, and set, respectively), just as the relational model provides but a single data type, the relation. In contrast, lower level languages, like PL/I, have a variety of data types at the programmer's disposal.

### Conclusions

In the past, commercial database systems have almost uniformly been based on the network or hierarchical model, because the emphasis of such systems has been on the maintenance of large databases, and these models lend themselves most easily to the necessary efficient implementation. However, it is felt that the relational model will receive the bulk of attention in the future for two reasons. First, it is becoming clearer that the same concepts used to design large databases apply as well to small and medium scale databases, and there are many more small (i.e., thousands of records rather than millions) databases than large ones. With small databases, the ease of use inherent in the relational model assumes increased importance.

Second, many of the apparent inefficiencies of the relational model can be eliminated. Chapter 6 discusses some of the optimization techniques for relational data manipulation languages that allow these languages to use time efficiently. Research aimed at producing good physical implementations of relations is also underway. We shall discuss some of the progress connected with IBM's System R in Section 4.5, and Section 4.7 discusses Query-by-Example, an existing commercial system suitable for medium size databases. Just as general-purpose very high level languages such as APL are winning converts as the rather difficult optimization problems associated with these languages gradually are solved, we suspect that the relational model will become more attractive as the optimization problems associated with this model are likewise solved.

### Exercises

- 3.1. In Fig. 3.16 we see the entity-relationship diagram of an insurance company. The keys for EMPLOYEES and POLICIES are EMP# and P#, respectively; SALESMEN are identified by their isa relationship to EMPLOYEES. Represent this diagram in the (a) relational (b) network (c) hierarchical models.

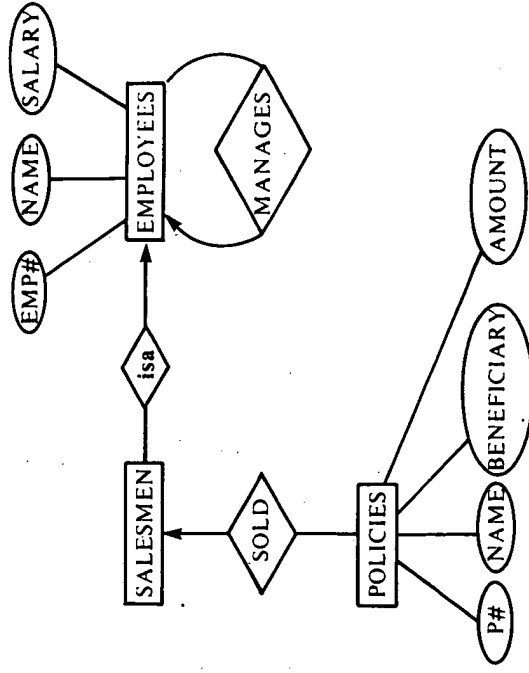


Fig. 3.16. An insurance company database.

- 3.2. Figure 3.17 shows a genealogy database, with key attributes NAME and LIC#. Represent this diagram in the (a) relational (b) network and (c) hierarchical models.

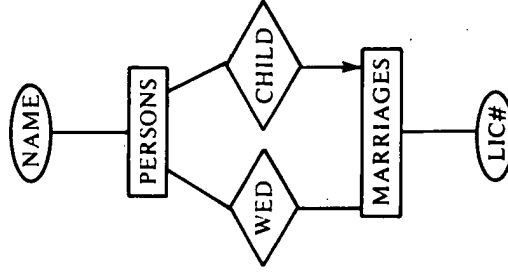


Fig. 3.17. A genealogy database.

# ***Design your own database***

Dr Lorna Scammell, University Database adviser - [Lorna.Scammell@ncl.ac.uk](mailto:Lorna.Scammell@ncl.ac.uk) - [contact details](#)

---

## | Summary of the characteristics of a relational database |

### **A series of steps identify what your database is about.**

The best way to do this is to work through the stages and draw a diagram (in pencil, as there is a lot of trial and error with this). Do the subjects and the relationships first - and then add the characteristics. The idea is that the subjects (entities) become the tables in the database and the characteristics become the fields. The relationships allow you to decide how the tables will link together. So when you have a good diagram ( sometimes called a model or logical design), you can create the database from that design. There are sometimes problems which need thought and discussion.

- 1 Decide on what subjects (entities) are covered by the database**
  - 2 Decide how the subjects are related to one another**
  - 3 Decide on what characteristics (attributes) the subjects have**
  - 4 Derive the database tables from the design**
- 

### **Summary of the characteristics of a relational database**

There is a theoretical background to relational databases. For the purposes of setting up your own database - it is enough to look at some examples and also appreciate that there are some restrictions on what you can do.

These can be summarised as:-

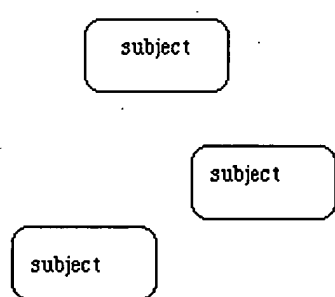
- Relational databases consist of one or more tables; these can be 'joined' by the database software in queries.
- Each table consists of rows and fields.
- Each table is about one aspect (or subject) of the database. Thus contexts and finds are different subjects and are in different tables.
- Each row corresponds to one instance of the subject of the table. Thus each row is about one context.
- Each row must be unique. This is a logical result of the row being about one instance. If you have duplicate rows, the results of searching are unpredictable.
- Each field corresponds to a variable and is named to indicate its role. For example finds have a name and a size/weight.
- Each cell (where the fields and rows intersect) contains only one value. This is important because otherwise it is not possible properly to search. Using a relational database, if you find a need for two values per cell - the design has to be altered.
- If fields in different tables have the same range of values and are thus about the same object, there is an association between the fields and thus the tables - they are called 'keys'. The rows corresponding to matching values can be retrieved from different tables.

## 1. Decide on what subjects (entities) are covered by the database

In designing a database it is useful to start at a very high level of generalisation.

- Look at what the subjects the database is about rather than the conclusions that you want to find
- Think about the subjects (and thus the data) separately from practical considerations, such as who is going to enter the data
- Think about the subject independently of any particular database software, or even of computing at all.
- Avoid getting confused between the overview of the data and details of implementation.

This section works through the construction of an appropriate diagram for your own problem similar to those used in the example data. The diagram becomes the basis for the tables and fields.

<h3><b><i>The entities</i></b></h3>	
<p>The technique is called an entity-attribute-relationship model.</p>	
<p>Entities are the things that hold particular interest for you in your database - you can think of them as the 'subjects' to be covered - but 'entity' is a more accurate term. They are a classification of things and should have a precise definition. They can be concrete things or abstractions. It is important to identify them because they are potentially the tables in the database.</p>	
<p>An entity is represented diagrammatically by a box with rounded corners and a name written in the singular. The name for an entity is one that represents a <i>class</i> of things in general - not specific instances.</p>	
<p>It may not be immediately obvious what the exact subjects of your problem might be.</p> <p>The best approach is to take a large sheet of paper and draw a box for each of the entities and put a name for each in the box.</p> <p>If you have a complex problem, choose the most important entities to begin with or you will be overwhelmed. This should be a quick sketch to start with.</p>	

## 2. Decide how the entities are related to one another

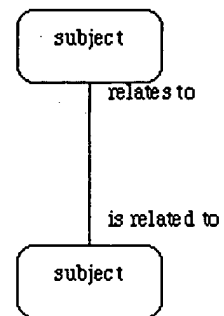
<h3><b><i>the relationships</i></b></h3>	
<p>A relationship is a significant association between two entities. It is represented by a line that joins two entity boxes. Each relationship has two ends, for each of which there is a name.</p> <p>(Relationships can also be mandatory or optional but this aspect is not included here as it is usually not as important in research databases as it is in the commercial world).</p>	

Use a simple word that encapsulates the relationship you see between the entities.

Be willing to alter the diagram and reposition the boxes and lines (thus the need for pencils and rubbers)

Simplify by concentrating on the important entities - the rest can be added as you come to understand the problem better.

There may well be more than one solution when the problem is a complex one.



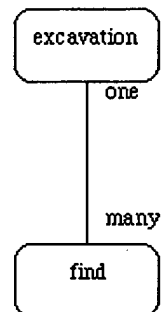
## Degree of relationship

### ***One-to-many relationships***

This means how many of one entity can be related to how many of the other(s). Ask yourself how many of each entity there can be.

In this kind of relationship one of the entities can be related many times to another entity. For example, one excavation can contain many finds; one person can commit many crimes.

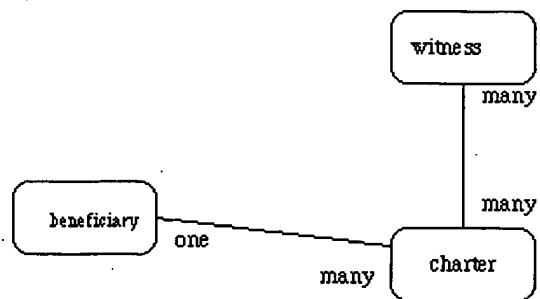
One to many relationships are common and are well handed by relational databases.



### ***Many-to-many relationships***

Here there can be many occurrences of one entity related to many occurrences of another entity. For example there can be many witnesses to each charter and many charters can be witnessed by any witness.

These relationships can be difficult to identify and to resolve. As a rule of thumb, if some relationship puzzles you, there is a good chance that there is a many to many relationship.



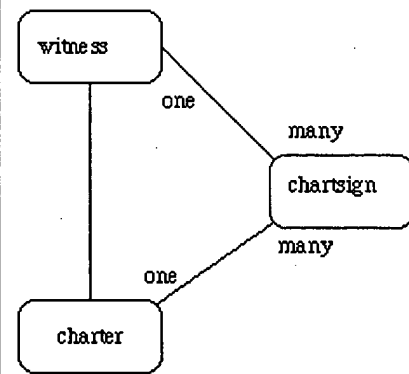
### ***Many-to-many relationships - a solution***

Many-to-many relationships are dealt with by the many-to-many relationship being decomposed into two relationships, with a newly created entity intervening between the original entities. Thus a many-to-many relationship can be replaced by an entity which has a many-to-one



relationship with each of the original entities, as illustrated by the example databases.

Here the idea of the 'chartsign' is that at the signing of any one charter there is one charter but many witnesses. Any witness can sign a number of charters.



The degree of the relationship is often difficult to identify in the early stages but if they are missed, it will show up when you come to create a real database and enter data. You then need to go back and look at the diagrams again, so it is best to spot these at the outset.

### 3. Decide on the characteristics (attributes) of the entities

#### Attributes

The next step is to add details about each entity. It is often useful to write them on the diagram so that they are clearly associated with the entity, although this can make the diagrams look rather like spiders webs.

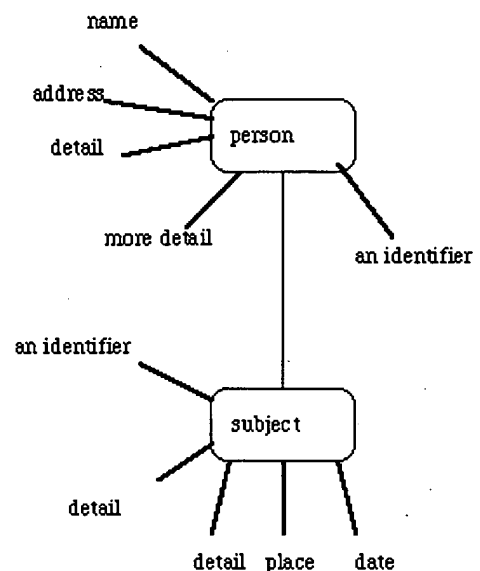
#### *the attributes*

Attributes are the details about the state of an entity; they are any description of an entity; they are properties of things we want to know about entities. For example, we might want some information about a person, such as their name or date of birth.

Attributes represent the data that is to be kept about the entities - and thus become the rows in the tables.

The diagrams tend to look a bit odd at this stage and you may have to try several approaches. As a guide, there will be between two and eight attributes for each entity - look at the diagrams again if you have more - you may have missed something.

At this stage it is also useful to identify how the attributes to be related to one another can be linked by a unique 'key'.



Researchers are often more aware of the importance of the characteristics of the things they are dealing with than the overall picture so it is often easier to identify attributes than entities. However, attributes and entities can be easily muddled because on different occasions the same items may be treated in different ways.

An attribute becomes an entity when it is a thing of significance in its own right, with its own relationships and

attributes. Equally, entities may act as attributes in some circumstances.

Make sure that your definitions apply equally accurately to every possible instance - not just the normal case.

### Where next?

| [Create tables](#) | [References about relational database design](#) | [Back to introduction](#) |

The database adviser is always willing to look at data models - [contact details](#)

---

### And

### Document what you have done:

Keep a note of your diagrams; describe the entities, attributes and relationships so that you can recall what assumptions you have made - in some cases this is important to the conclusions you can validly draw from the data.

---

*Lorna.Scammell@ncl.ac.uk*

*October 2nd 1997 / moved and checked links - August 2001 / Feb / April 2003 /*

---



filter

Search

☒ Dictionary
 ☐ Thesaurus
 ☐ Encyclopedia
 ☐ Web
[Home](#)
 Premium: [Sign up](#) | [Login](#)
**Technical Analysis**

Get high-quality on-line financial charts at StockCharts.com.

**Free Software Download**

TeleChart 2005 + 25 Year Databank Voted Best Stock Charts since 1993

**Trading Secrets Book Free**

Get the book "Secrets of Successful Traders" free from MetaStock

**Technical Analysis**

Get Real-Time, Accurate Data Free Charting &amp; An

[Ads by Google](#)[Advertise](#)

## ADVERTISEMENT

[Dictionary](#) - [Thesaurus](#) - [Encyclopedia](#) - [Web](#)
Top Web Results for "filter"

## ADVERTISEMENT

**9 entries found for *filter*.**

**fil·ter** **Pronunciation Key** (fĭl'ĭtər)  
*n.*

- a. A porous material through which a liquid or gas is passed in order to separate the fluid from suspended particulate matter.
- b. A device containing such a material, especially one used to extract impurities from air or water.
2.
  - a. Any of various electric, electronic, acoustic, or optical devices used to reject signals, vibrations, or radiations of certain frequencies while allowing others to pass.
  - b. A colored glass or other transparent material used to select the wavelengths of light allowed to reach a photosensitive material.
3. Computer Science. A program or routine that blocks access to data that meet a particular criterion: *a Web filter that screens out vulgar sites.*

*v.* **fil·tered, fil·ter·ing, fil·ters**

*v.* **tr.**

1. To pass (a liquid or gas) through a filter.
2. To remove by passing through a filter: *filter out impurities.*
3. Computer Science. To use a filter to block access to (a website or Web content).

v. *intr.*

1. To pass through or as if through a filter: *Light filtered through the blinds.*
2. To come or go gradually and in small groups: *The audience filtered back into the hall.*

[Middle English *filtre*, from Old French, from Medieval Latin *filtrum*, of Germanic origin. See *pel*-<sup>5</sup> in Indo-European Roots.]

**fil'ter·er** *n.*

**fil'ter·less** *adj.*

[Download Now or Buy the Book]

Source: *The American Heritage® Dictionary of the English Language, Fourth Edition*  
Copyright © 2000 by Houghton Mifflin Company.  
Published by Houghton Mifflin Company. All rights reserved.

**fil·ter** (fĭl'tər)

*n.*

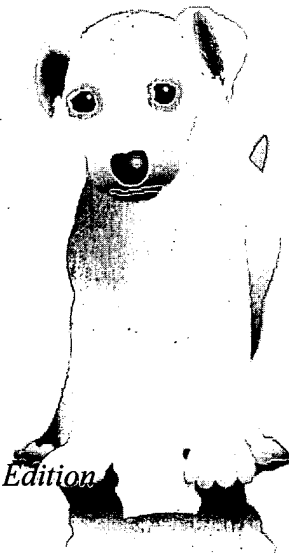
1. A porous material through which a liquid or gas is passed in order to separate the fluid from suspended particulate matter.
2. A device containing such a substance.
3. Any of various electric, electronic, acoustic, or optical devices used to reject signals, vibrations, or radiations of certain frequencies while passing others.
4. A translucent screen, used in both diagnostic and therapeutic radiology, that permits the passage of rays having desirable levels of energy.
5. A device used in spectrophotometric analysis to isolate a segment of the spectrum.

v. **fil·tered**, **fil·ter·ing**, **fil·ters**

1. To pass a liquid or gas through a filter.
2. To remove by passing through a filter.

**CLICK  
TO SAVE**  
on your pet's  
medications.

Play with me!



**1-800-PetMeds**

**Related ads:**

- [RF Notch Filter](#)
- [Band Pass Filter](#)
- [Bessel Filter](#)
- [Active Filter Design](#)
- [Anti Spam Filter](#)

3. To pass through or as if through a filter.

---

**fil'ter·er** *n.*

**fil'ter·less** *adj.*

Source: *The American Heritage® Stedman's Medical Dictionary*  
 Copyright © 2002, 2001, 1995 by Houghton Mifflin Company. Published by Houghton Mifflin Company.

---

## Filter

A set of criteria used to help an investor narrow down which financial instruments or conditions of financial instruments are the most profitable.

### *Investopedia Commentary*

Most beginner investors feel overwhelmed by the large number of financial products available in each type of market. Filters help narrow down the investor's or trader's search, so the decision of which securities to trade is less complicated. Filters are not limited to finding companies by stock screeners technical analysis tools can also help an investor or trader find a particular financial instrument. For example, if a trader wants to find financial instruments that are trading above their 200-day moving average, he or she can use a filter to find such instruments.

### *Related Links*

[Getting To Know Stock Screeners](#)

[Introduction to Types of Trading: Technical Traders](#)

[Introduction to Types of Trading: Fundamental Traders](#)

See also: [Moving Average](#), [Simple Moving Average](#), [Stock Screener](#), [Technical Analysis](#)

Source: *Investopedia.com*. Copyright © 1999-2005 - All rights reserved. Owned and Operated by *Investopedia Inc.*

---

Main Entry: <sup>2</sup>**filter**

Function: *verb*

Inflected Forms: **fil·tered**; **fil·ter·ing** /-t (&-) ri [ng] /

*transitive senses*

**1** : to subject to the action of a filter

**2** : to remove by means of a filter **filter** *intransitive senses*

: to pass or move through or as if through a filter

Source: *Merriam-Webster's Medical Dictionary*, © 2002 Merriam-Webster, Inc.

---

Main Entry: <sup>1</sup>**fil·ter**

Pronunciation: 'fil-t&r

Function: *noun*

**1** : a porous article or mass (as of paper or sand) through which a gas or liquid is passed to separate out matter in suspension

**2** : an apparatus containing a filter medium

**3 a** : a device or material for suppressing or minimizing waves or oscillations of certain frequencies (as of electricity, light, or sound) **b** : a transparent material (as colored glass) that absorbs light of certain wavelengths or colors selectively and is used for modifying light that reaches a sensitized photographic material called also *color filter*

Source: *Merriam-Webster's Medical Dictionary*, © 2002 Merriam-Webster, Inc.

## filter

n 1: device that removes something from whatever passes through it 2: an electrical device that alters the frequency spectrum of signals passing through it v 1: remove by passing through a filter; "filter out the impurities" [syn: filtrate, strain, separate out, filter out] 2: pass through; "Water permeates sand easily" [syn: percolate, sink in, permeate] 3: run or flow slowly, as in drops or in an unsteady stream; "water trickled onto the lawn from the broken hose"; "reports began to dribble in" [syn: trickle, dribble]

Source: *WordNet* ® 2.0, © 2003 Princeton University

## filter

1. (Originally Unix, now also MS-DOS) A program that processes an input data stream into an output data stream in some well-defined way, and does no I/O to anywhere else except possibly on error conditions; one designed to be used as a stage in a pipeline (see plumbing). Compare sponge.

2. (functional programming) A higher-order function which takes a predicate and a list and returns those elements of the list for which the predicate is true. In Haskell:

```
filter p [] = []
filter p (x:xs) = if p x then x : rest else rest
where
rest = filter p xs
```

See also filter promotion.

[Jargon File]

Source: *The Free On-line Dictionary of Computing*, © 1993-2005 Denis Howe

**filter**

n. [very common; orig. Unix, now also in MS-DOS] A program that processes an input data stream into an output data stream in some well-defined way, and does no I/O to anywhere else except possibly on error conditions; one designed to be used as a stage in a 'pipeline' (see plumbing). Compare sponge.

Source: Jargon File 4.2.0

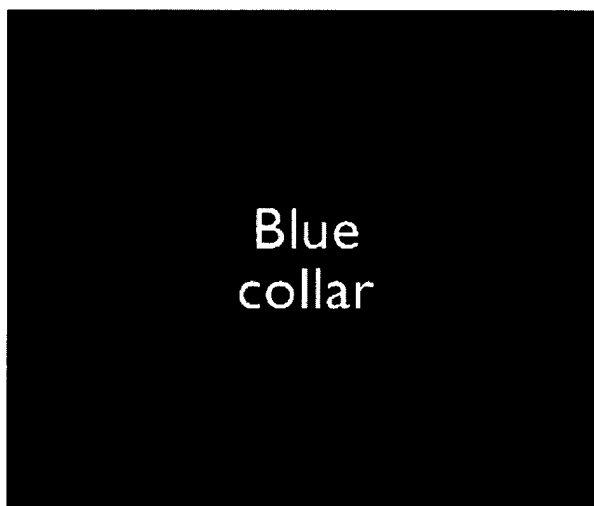
**filter**

filter: in CancerWEB's On-line Medical Dictionary

Source: On-line Medical Dictionary, © 1997-98 Academic Medical Publishing & CancerWEB

**Perform a new search, or try your search for "filter" at:**

- [Amazon.com](#) - Shop for books, music and more
- [HighBeam Research](#) - 32 million documents from leading publications
- [Merriam-Webster](#) - Search for definitions
- [Reference.com](#) - Encyclopedia Search
- [Reference.com](#) - Web Search powered by Google
- [Thesaurus.com](#) - Search for synonyms and antonyms



ADVERTISEMENT

Get the **FREE Dictionary.com Toolbar** for your browser now!

From the makers of Dictionary.com

Copyright © 2006, [Lexico Publishing Group, LLC](#). All rights reserved.

[About Dictionary.com](#) | [Privacy Policy](#) | [Terms of Use](#) | [Link to Us](#) | [Help](#) | [Contact Us](#)



parse

Search

☒ Dictionary ☐ Thesaurus ☐ Encyclopedia ☐ Web
[Home](#)
 Premium: [Sign up](#) | [Login](#)


**SHE MARRIED HIM??!!**  
**AND THEY'VE GOT 7 KIDS??**  
**classmates.com**



**Find Your Old  
School Here**

- City  
 - State

ADVERTISEMENT

[Dictionary](#) - [Thesaurus](#) - [Encyclopedia](#) - [Web](#)
Top Web Results for "parse"

ADVERTISEMENT

## 5 entries found for *parse*.

**parse** **Pronunciation Key** (pärs)

v. **parsed, pars·ing, pars·es**

v. *tr.*

1. To break (a sentence) down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part.
2. To describe (a word) by stating its part of speech, form, and syntactical relationships in a sentence.
3.
  - a. To examine closely or subject to detailed analysis; especially by breaking up into components: "What are we missing by parsing the behavior of chimpanzees into the conventional categories recognized largely from our own behavior?" (Stephen Jay Gould).
  - b. To make sense of; comprehend: *I simply couldn't parse what you just said.*
4. Computer Science. To analyze or separate (input, for example) into more easily processed components.

v. *intr.*



To admit of being parsed: *sentences that do not parse easily.*

[Probably from Middle English *pars*, *part of speech*, from Latin *pars* (*ōrātiōnis*), *part (of speech)*. See *per-*<sup>2</sup> in Indo-European Roots.]

**pars** *er n.*

[Download Now or Buy the Book]

Source: *The American Heritage® Dictionary of the English Language, Fourth Edition*

Copyright © 2000 by Houghton Mifflin Company.

Published by Houghton Mifflin Company. All rights reserved.

**parse**

parse was Word of the Day on June 3, 2001.

Source: *Dictionary.com Word of the Day*

**parse**

v : analyze syntactically by assigning a constituent structure to (a sentence)

Source: *WordNet® 2.0, © 2003 Princeton University*

**parse**

[from linguistic terminology] vt. 1. To determine the syntactic structure of a sentence or other utterance (close to the standard English meaning). "That was the one I saw you." "I can't

parse that." 2. More generally, to understand or comprehend. "It's

very simple; you just kretch the glims and then aos the zotz." "I can't parse that." 3. Of fish, to have to remove the bones yourself. "I object to parsing fish", means "I don't want to get a whole fish, but a sliced one is okay". A 'parsed fish' has been deboned. There is some controversy over whether 'unparsed' should

mean 'bony', or also mean 'deboned'.

Ads by Google

#### **Visual Parser Generator**

Build your own parsers with ProGrammar. Free technical support.

[www.programmar.com](http://www.programmar.com)

#### **Top Loan Companies**

Free Review of the Top Loan and Debt

Consolidation Sources.

Consumer-Protection-Company.com

#### **Win A \$10,000**

##### **Scholarship**

Registration Is Free. Fast & Easy Takes 11

Seconds. Win \$10,000

[10000Scholarship.com](http://10000Scholarship.com)

#### **English Sentences**

##### **Checker**

Check Grammar,

Thesaurus, Spelling Fix &

Enhance Your Text- Free

Trial

[www.WhiteSmoke.com](http://www.WhiteSmoke.com)

#### **Win A \$10,000**

##### **Scholarship**

If You Have 15 Seconds,

Enter To Win A \$10,000

College Sweepstakes

[www.DirectScholar.com](http://www.DirectScholar.com)

#### **Related ads:**

- [Visual Parse](#)
- [Parser Generator](#)
- [Simple API for XML Parsing](#)
- [Resume Parsing](#)
- [Parse Email](#)

Source: *Jargon File 4.2.0*

**parse**

PARSE: in Acronym Finder

Source: *Acronym Finder*, © 1988-2004 Mountain Data Systems

**Perform a new search, or try your search for "parse" at:**

- [Amazon.com](#) - Shop for books, music and more
- [HighBeam Research](#) - 32 million documents from leading publications
- [Merriam-Webster](#) - Search for definitions
- [Reference.com](#) - Encyclopedia Search
- [Reference.com](#) - Web Search powered by Google
- [Thesaurus.com](#) - Search for synonyms and antonyms



**MOTOROLA RAZR V3**  
**\$149.99**  
FULL-AUDIO RINGTONES,  
GRAPHICS & GAMES!  
**Own It Now**  
**cingular**  
2-YEAR SERVICE AGREEMENT AND  
OTHER RESTRICTIONS APPLY.

ADVERTISEMENT

Get the **FREE Dictionary.com Toolbar** for your browser now!

From the makers of Dictionary.com

Copyright © 2006, Lexico Publishing Group, LLC. All rights reserved.

[About Dictionary.com](#) | [Privacy Policy](#) | [Terms of Use](#) | [Link to Us](#) | [Help](#) | [Contact Us](#)

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**